SIEMENS



西门子工业自动化系列教材

# 西门子S7-300/400 PLC 编程技术及工程应用

陈海霞 柴瑞娟 任庆海 孙承志 编著






附赠电子教案

http://www.cmpedu.com




附软件：

Step 7 V5.4 SP3 + S7-PLCSIM V5.4 SP1+

用户手册 + 应用文档 + 电子课件 + 程序





机械工业出版社

CHINA MACHINE PRESS

VISIT…

LANZAROTE Caliente.COM

西门子工业自动化系列教材

# 西门子 S7-300/400 PLC 编程技术及工程应用

陈海霞　柴瑞娟
任庆海　孙承志　编著



机 械 工 业 出 版 社

西门子 S7-300 及 S7-400 是面向系统解决方案的通用型 PLC，其应用相当广泛。

本书主要分为六大部分：第一部分是对 S7-300 及 S7-400 的系统概述，介绍了 S7-300 和 S7-400 的工作原理、硬件结构、安装配置及模块特性，使读者对 PLC 系统的体系架构有一定的了解；第二部分介绍了 STEP 7 的编程环境、硬件组态及调试方法；第三部分介绍了基于 IEC61131-1 的编程语言及先进的编程技术思想：顺序功能图（S7 Graph）和状态图（S7 HiGraph）；第四部分介绍了组织块和系统功能块的作用；第五部分介绍工业网络通信的基本方法和人机界面的通信；第六部分介绍了工程设计步骤和两个工程实例。本书的编写宗旨是：通过大量的实验案例和真实的工程实例使学习和实践能融会贯通；通过实用编程技术的介绍，提供易于交流的平台和清晰的编程思路。随书附赠学习光盘，内容包括 STEP 7 V5.4 编程软件、程序、参考课件和软硬件参考手册。

本书可供工程技术人员自学和参考，也可作为高等院校本科自动化及相关专业的参考教材。

**图书在版编目(CIP)数据**

西门子 S7-300/400 PLC 编程技术及工程应用/陈海霞等编著. —北京：机械工业出版社，2011.12（2013.1重印）
西门子工业自动化系列教材
ISBN 978-7-111-36617-1

Ⅰ.①西… Ⅱ.①陈… Ⅲ.①可编程序控制器-程序设计-教材
Ⅳ.①TM571.6

中国版本图书馆 CIP 数据核字(2011)第 244266 号

机械工业出版社(北京市百万庄大街 22 号　邮政编码 100037)
策划编辑：时　静
责任编辑：时　静
责任印制：张　楠
唐山丰电印务有限公司印刷

2013 年 1 月第 1 版 · 第 2 次印刷
184mm×260mm · 26 印张 · 643 千字
4001-8000　册
标准书号：ISBN 978-7-111-36617-1
　　　　　ISBN 978-7-89433-244-8（光盘）
定价：59.80　元(含 1CD)

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
社服务中心：(010)88361066
销售一部：(010)68326294
销售二部：(010)88379649
读者购书热线：(010)88379203

网络服务
门户网：http://www.cmpbook.com
教材网：http://www.cmpedu.com

**封面无防伪标均为盗版**

# 序

从 1969 年美国数字公司研制出的第一代可编程序控制器，到现在为止可编程序控制器已经有了第 5 代产品了，随之而来的是可编程技术的日益成熟。可编程序控制器已逐步发展成以微处理器为核心，集计算机技术、自动化控制技术及通信技术于一体的工业控制装置。与网络技术及组态软件技术的配合，可编程序控制器的应用如虎添翼。

可编程序控制器在工业领域的应用非常广泛，既有单机作为继电器逻辑电路的替代品，又有作为控制设备的核心部件。随着自动化程度的提高，它既可以作为现场控制的部件，又可以作为现场更高一级管理的控制部件。随着网络技术的发展，作为成熟技术，可编程序控制器已被广泛应用到机械、冶金、化工、石化、水泥、制药等各个领域，极大地提高了劳动生产率和自动化程度。与 DCS、IPC 技术形成三足鼎立之势，占据着市场的最大份额。

西门子是中国多个业务领域的领先工业解决方案供应商，在生产自动化、过程自动化、驱动、低压控制以及安装技术方面提供了各类创新、可靠、高效和优质的产品。并全面提供系统的解决方案和服务。产品涵盖范围广，在信息与通信、自动化与控制、电力、交通、医疗、照明等各个行业领域处于领先优势。

西门子 PLC 技术能够不断融入新技术、新方法、推陈出新，并进一步在 e－制造和 e－控制技术方面进行新的突破，新的产品不断涌现，为各种各样的自动化控制设备提供了非常可靠的控制应用。

目前 PLC 产品在许多行业的大中型企业中均有应用，市场占有率很高，但是与其广泛的应用相比，介绍西门子公司 PLC 产品的应用书籍却比较少，影响了产品的宣传，尤其是在大、专院校的技术推广。《西门子 S7-300/400 PLC 编程技术及工程应用》一书详细介绍了 PLC 的几种先进的实用编程技术：如顺序功能图（Sequential Function Charts）和状态图（State Graph）等，其严格的编程思路能够避免设计系统的不完善性，提高了自动化工程师的工作效率，缩短了项目的开发周期。书中介绍的系统性编程技术在现有的 PLC 书籍中实属罕见，并且以实际工程项目为基础，详细阐述了工程设计步骤及内容，而在一般的 PLC 书籍中，基本没有工程应用的介绍。

希望《西门子 S7-300/400 PLC 编程技术及工程应用》一书能为更多的工程技术人员和广大的学生提供帮助和支持。使读者能够尽快掌握 PLC 的基础及应用技术，并尽快应用于工程设计中，缩短学习和工程应用之间的距离。

西门子（中国）有限公司高级副总裁

安晓杰

# 前　言

可编程序控制器（PLC）是计算机家族中的一员，它是为工业控制应用而设计制造的综合了计算机技术、自动控制技术和通信技术的一种新型的、通用的自动控制装置。PLC 以其功能强、可靠性高、使用灵活方便、易于编程以及适于在工业环境下应用等一系列优点，而成为工业控制领域中增长速度最迅猛的工业控制设备，PLC 已经成为现代工业自动化的三大支柱之一。

目前，PLC 产品大致可分为美国、欧洲国家、日本三大流派。据统计，德国西门子公司的 PLC 在我国 PLC 市场上的占有量已经超过 30%，特别是西门子公司推出的 S7-300/400 系列的 PLC 以其功能强大、性价比高等特点而深受国内用户的欢迎。为了使用户更易于了解并尽快地掌握西门子 S7-300/400 系列 PLC 的性能和特点，并更好地应用于实践中去，编写一本基于西门子 S7-300/400 系列 PLC 的书籍是十分必要的。

本书以德国西门子公司的 S7-300/400 系列 PLC 为主线，以 STEP 7 编程系统为平台，系统地介绍了 PLC 的基础理论、编程方法以及在工业中的应用等知识。新颖、实用、易读是本书的编写宗旨。为了便于教学和自学，本书还精心编写了大量的例题及其实现程序，而且每一个程序都用仿真软件 PLCSIM 或在 PLC 上做了验证。在本书的光盘中不仅包括例题的实现程序，还包括大量西门子 S7-300/400 系列 PLC 的参考资料。

在编写本书过程中，编者力求做到语言流畅、叙述清楚、讲解细致，所有内容都立足于实际应用和教学，并融入编者的经验和成果。本书详尽地叙述了 PLC 控制系统所需要的多种知识，全书共分为 11 章。前三章是基础知识，主要介绍了 PLC 的由来、特点和工作原理，S7-300/400 系列 PLC 的特点及功能，以及 STEP 7 编程软件的用法；第 4 章以不同的编程语言，通过大量的例题说明 PLC 的各种编程元件的使用；第 5 章较为全面地总结了 STEP 7 的调试方法；第 6 章详细介绍了先进的编程技术思想：顺序功能图（S7 Graph）和状态图（HiGraph）；第 7 章和第 8 章介绍了结构化编程方法以及组织块及系统功能的用法；第 9 章用几个实例详细讲解了工业网络的组网方法；第 10 章介绍了西门子人机界面的基本用法；第 11 章以两个具体的工程实例说明 PLC 控制系统的设计、实现及上位机的监控。其中陈海霞编写了第 1 ~5 章和第 8 章，柴瑞娟编写第 6 章和第 7 章，孙承志编写了第 9 章，任庆海编写最后两章，全书由陈海霞统稿。

西门子（中国）有限公司高级副总裁安晓杰先生在百忙之中为本书写序，在此编者表示感谢！本书在编写过程中得到了西门子（中国）有限公司杨大汉先生的大力支持，同时得到了三江学院西门子自动化示范实验室的硬件支持以及南京理工大学研究生的配合，再次表示衷心的感谢！

由于编者的水平有限，错误和疏漏在所难免，恳请读者不吝指正。

作者的 E－mail 地址 ：nj_chx@126. com。

编　者

2011 年 7 月 1 日

# 目　　录

序
前言
第 1 章　PLC 基础 …… 1
1.1　概述 …… 1
1.1.1　PLC 的发展史 …… 1
1.1.2　PLC 的主要特点 …… 2
1.1.3　PLC 的主要应用 …… 3
1.2　西门子 PLC 概述 …… 4
1.2.1　西门子“全集成自动化”概念 …… 4
1.2.2　西门子 PLC 产品 …… 6
1.2.3　S7-300 系列 PLC …… 6
1.2.4　S7-400 系列 PLC …… 7
1.2.5　S7-1200 系列 PLC …… 7
1.3　PLC 的组成 …… 8
1.3.1　PLC 的基本结构 …… 8
1.3.2　S7-300/400 系列 PLC 的组成 …… 8
1.4　PLC 的工作原理 …… 10
1.4.1　工作原理 …… 10
1.4.2　循环时间和响应时间 …… 12
习题 …… 12
第 2 章　S7-300/400 结构体系 …… 13
2.1　S7-300 的 CPU 模块 …… 13
2.1.1　CPU 的分类 …… 13
2.1.2　CPU 的面板 …… 14
2.1.3　CPU 的存储器 …… 15
2.2　S7-300 的信号模块 …… 15
2.2.1　数字量模块 …… 16
2.2.2　模拟量模块 …… 17
2.3　S7-300 的特殊模块 …… 19
2.3.1　通信处理模块 CP 34x …… 19
2.3.2　计数器模块 FM 350 和 CM 35 …… 19

2.3.3 位置控制与位置检测模块 FM 35x …… 20
2.3.4 闭环控制模块 FM 355 …… 20
2.3.5 称重模块 SIWAREX …… 20
2.4 硬件模块的安装 …… 21
2.4.1 安装导轨（RACK） …… 21
2.4.2 安装模块 …… 22
2.4.3 接线 …… 22
2.5 寻址 …… 23
2.5.1 存储区中的地址及格式 …… 23
2.5.2 基于槽编址的模块地址 …… 24
2.5.3 用户编址的模块地址 …… 25
习题 …… 25
第3章 STEP 7 的使用基础 …… 26
3.1 STEP 7 概述 …… 26
3.2 安装与卸载 STEP 7 …… 27
3.2.1 系统配置要求 …… 27
3.2.2 安装 STEP 7 …… 28
3.2.3 卸载 STEP 7 …… 28
3.3 SIMATIC 管理器 …… 28
3.4 硬件组态 …… 29
3.4.1 硬件组态步骤 …… 29
3.4.2 参数设置 …… 31
3.4.3 硬件组态目录的更新 …… 33
3.5 软件编程 …… 34
3.5.1 程序编辑器界面 …… 34
3.5.2 使用程序编辑器 …… 34
3.5.3 变量与符号 …… 35
3.6 硬件接口和下载 …… 37
3.6.1 硬件接口 …… 37
3.6.2 下载方法 …… 37
3.6.3 上传 …… 42
3.7 程序归档 …… 42
3.8 如何使用 STEP 7 软件的在线帮助 …… 43
3.8.1 查找某个关键字或功能 …… 43
3.8.2 了解某个逻辑块 FB/FC/SFB/SFC 的功能及管脚的定义 …… 43
3.8.3 应用方法 …… 43
习题 …… 43

第 4 章　编程语言 …… 44
4.1　概述 …… 44
4.2　STEP 7 编程语言的程序结构 …… 45
4.2.1　用户块 …… 45
4.2.2　系统块 …… 46
4.3　指令结构 …… 47
4.3.1　指令组成 …… 47
4.3.2　数据类型及存储区 …… 47
4.3.3　CPU 存储区 …… 50
4.3.4　寻址方式 …… 50
4.3.5　状态字和逻辑操作过程 …… 52
4.4　位逻辑指令 …… 54
4.4.1　位逻辑运算指令 …… 54
4.4.2　位操作指令 …… 58
习题Ⅰ …… 66
4.5　定时器与计数器指令 …… 68
4.5.1　定时器 …… 68
4.5.2　计数器 …… 81
习题Ⅱ …… 85
4.6　数据处理功能指令 …… 86
4.6.1　装载和传输指令 …… 86
4.6.2　比较指令 …… 88
4.6.3　转换指令 …… 94
4.6.4　移位和循环移位指令 …… 97
4.6.5　累加器操作和地址寄存器指令 …… 100
4.7　数据运算指令 …… 103
4.7.1　整数算术运算 …… 103
4.7.2　浮点数算术运算 …… 104
4.7.3　字逻辑运算指令 …… 105
4.8　控制指令 …… 106
4.8.1　逻辑控制指令 …… 107
4.8.2　程序控制指令 …… 112
4.8.3　主控继电器指令 …… 114
习题Ⅲ …… 115
4.9　应用实例 …… 116
4.9.1　常用指令的综合用法 …… 116
4.9.2　ET200M 的使用 …… 124

4.9.3 变频器的使用 …… 128
第5章 调试方法 …… 136
5.1 利用LED指示灯调试 …… 136
5.2 硬件组态的调试 …… 137
5.2.1 下载硬件组态时的调试 …… 137
5.2.2 建立在线连接 …… 139
5.2.3 利用“Module Information”工具调试 …… 139
5.2.4 硬件组态窗口中信号的检测与修改 …… 141
5.2.5 诊断符号 …… 142
5.3 离线/在线程序块的比较 …… 142
5.4 利用程序状态调试 …… 144
5.4.1 监控程序状态的前提 …… 144
5.4.2 监视程序的状态 …… 144
5.4.3 STL程序的单步与断点调试 …… 145
5.5 利用变量表调试 …… 146
5.5.1 变量表的功能 …… 146
5.5.2 建立变量表 …… 147
5.5.3 变量表的使用 …… 147
5.6 利用“诊断缓冲区”调试 …… 150
5.7 参考数据（Reference Data） …… 154
5.7.1 参考数据的生成和显示方式 …… 154
5.7.2 参考数据表的种类 …… 154
5.7.3 在程序中快速查找地址的位置 …… 156
5.8 结构化程序的调试 …… 157
5.9 S7-PLCSIM的应用 …… 157
5.9.1 S7-PLCSIM介绍 …… 157
5.9.2 S7-PLCSIM的使用方法 …… 158
5.9.3 S7-PLCSIM的调试应用举例 …… 160
5.9.4 仿真PLC与真实PLC的区别 …… 161
习题 …… 162
第6章 编程技术 …… 163
6.1 控制系统的基本设计步骤 …… 164
6.1.1 分析和描述任务 …… 164
6.1.2 确定控制策略 …… 164
6.1.3 决定运行方式 …… 164
6.1.4 控制系统的调试 …… 165
6.2 编程技术基础 …… 165

6.2.1 程序设计举例 …… 166
6.2.2 编程要求 …… 169
6.3 控制系统分析方法及系统建模 …… 170
6.3.1 控制系统分析方法 …… 170
6.3.2 系统建模 …… 171
6.3.3 工程实例 …… 172
6.4 顺序功能图（SFC） …… 176
6.4.1 概述 …… 176
6.4.2 顺序功能图的绘制方法 …… 177
6.4.3 运用顺序功能图思想的编程方法 …… 181
6.4.4 具有多种工作方式系统的顺序功能图的编程方法 …… 192
习题Ⅰ …… 201
6.4.5 MPS 工作站的设计 …… 203
6.4.6 GRAPH 编程 …… 215
6.5 状态图（State Graph） …… 224
6.5.1 状态图简介 …… 224
6.5.2 状态图的建立方法及状态图的程序实现 …… 225
6.5.3 状态图应用实践 …… 231
习题Ⅱ …… 246
第7章 结构化编程 …… 247
7.1 概述 …… 247
7.1.1 程序设计方法 …… 247
7.1.2 块的含义及调用 …… 248
7.1.3 块的结构 …… 248
7.2 功能和功能块编程及调用举例 …… 249
7.2.1 功能编程及举例 …… 250
7.2.2 功能块编程及举例 …… 253
7.3 FC 和 FB 程序设计实例 …… 254
7.3.1 任务描述 …… 254
7.3.2 建立符号表 …… 256
7.3.3 生成电动机 FB …… 257
7.3.4 生成阀门 FC …… 259
7.3.5 生成 OB1 …… 260
习题 …… 265
第8章 组织块及系统功能的使用 …… 266
8.1 组织块 …… 266
8.2 循环处理的主程序 OB1 …… 267

8.3　日期时间中断组织块（OB10 ~ OB17） …… 268
8.3.1　概述 …… 268
8.3.2　应用方法 …… 269
8.3.3　应用实例 …… 270
8.4　延时中断组织块（OB20 ~ OB23） …… 272
8.4.1　概述 …… 272
8.4.2　应用方法 …… 273
8.4.3　应用实例 …… 274
8.5　循环中断组织块（OB30 ~ OB38） …… 275
8.5.1　概述 …… 275
8.5.2　应用方法 …… 276
8.5.3　应用实例 …… 277
8.6　硬件中断组织块（OB40 ~ OB47） …… 278
8.6.1　概述 …… 278
8.6.2　应用方法 …… 279
8.6.3　应用实例 …… 279
8.7　异步错误组织块 …… 281
8.7.1　时间错误处理组织块（OB80） …… 282
8.7.2　电源故障处理组织块（OB81） …… 282
8.7.3　诊断中断组织块（OB82） …… 282
8.7.4　机架故障组织块（OB86） …… 285
8.7.5　通信错误组织块（OB87） …… 288
8.8　起动组织块（OB100 ~ OB102） …… 288
8.9　同步错误组织块 …… 290
8.9.1　编程故障组织块（OB121） …… 290
8.9.2　I/O 访问故障组织块（OB122） …… 292
8.10　系统功能 …… 292
习题 …… 302
第 9 章　工业网络通信 …… 303
9.1　概述 …… 303
9.2　MPI 通信 …… 305
9.2.1　简介 …… 305
9.2.2　通信分类 …… 306
9.2.3　MPI 通信实例 …… 306
9.3　PROFIBUS 现场总线通信 …… 315
9.3.1　简介 …… 315
9.3.2　协议类型分类 …… 316

9.3.3 PROFIBUS-DP 通信及分类 …… 317
9.3.4 PROFIBUS-DP 通信实例 …… 317
9.4 工业以太网通信 …… 328
9.4.1 简介 …… 328
9.4.2 多台 S7－300 之间的 IE 通信 …… 328
第 10 章 西门子人机界面技术 …… 337
10.1 人机界面简介 …… 337
10.1.1 人机界面的基本概念 …… 337
10.1.2 人机界面的分类 …… 337
10.1.3 人机界面的功能 …… 337
10.2 基于触摸屏的监控网络 …… 338
10.2.1 触摸屏概述 …… 338
10.2.2 组态软件 WinCC Flexible 基础 …… 341
10.2.3 WinCC Flexible 过程通信 …… 341
10.2.4 应用举例 …… 343
10.3 基于 PC 的工业监控网络 …… 347
10.3.1 工控机概述 …… 347
10.3.2 组态软件 WinCC 基础 …… 347
10.3.3 WinCC 过程通信 …… 348
10.3.4 WinCC 通信组态 …… 350
第 11 章 PLC 在实际工程中的应用 …… 354
11.1 PLC 控制系统的设计 …… 354
11.1.1 设计原则 …… 354
11.1.2 设计内容 …… 354
11.1.3 设计步骤 …… 355
11.1.4 硬件设计 …… 357
11.1.5 软件设计 …… 361
11.1.6 PLC 控制系统的抗干扰设计 …… 362
11.2 系统调试与检查 …… 363
11.2.1 系统调试步骤 …… 363
11.2.2 系统调试方法 …… 364
11.3 交流电动机正、反转控制的工程应用方法 …… 364
11.3.1 工程应用基础 …… 365
11.3.2 控制原理 …… 366
11.4 闸门自动监控系统工程实例 …… 370
11.4.1 项目概况和要求 …… 370
11.4.2 系统总体设计 …… 371

11.4.3 PLC 模块及其他设备的选型 …… 372
11.4.4 控制原理图及设备接线图的设计 …… 373
11.4.5 设备组柜与接线工作 …… 380
11.4.6 PLC 硬件组态 …… 380
11.4.7 软件编程设计与调试 …… 383
11.4.8 上位机软件设计 …… 387
11.4.9 系统联调 …… 388
11.5 某钢厂大电炉水处理自动化监控系统工程实例 …… 388
11.5.1 项目概况和要求 …… 388
11.5.2 系统总体设计 …… 389
11.5.3 PLC 模块及其他设备的选型 …… 390
11.5.4 控制原理图及设备接线图的设计 …… 391
11.5.5 设备组柜与接线工作 …… 392
11.5.6 PLC 硬件组态 …… 394
11.5.7 软件编程设计与调试 …… 395
11.5.8 上位机软件设计 …… 398
11.5.9 系统联调 …… 398
**参考文献** …… 402

# 第 1 章　PLC 基础

## 1.1　概述

### 1.1.1　PLC 的发展史

在 20 世纪 60 年代，汽车生产流水线的自动控制系统基本上都是由继电器控制装置构成的。当时汽车的每一次改型都直接导致继电器控制装置的重新设计和安装。随着生产的发展，汽车型号更新的周期愈来愈短，这样，继电器控制装置就需要经常地重新设计和安装，十分费时、费工和费料，甚至阻碍了更新周期的缩短。为了改变这一现状，美国通用汽车公司在 1969 年公开招标，要求用新的控制装置取代继电器控制装置，并提出了十项招标指标，要求编程方便、现场可修改程序、维修方便、采用模块化结构等。1969 年，美国数字设备公司（DEC）研制出第一台 PLC，在美国通用汽车自动装配线上试用，获得了成功。

可编程序控制器（Programmable Controller）是计算机家族中的一员，是为工业控制应用而设计制造的。早期的可编程序控制器称为可编程逻辑控制器（Programmable Logic Controller），简称 PLC，它主要用来代替继电器实现逻辑控制。随着技术的发展，这种装置的功能已经大大超过了逻辑控制的范围。为了控制机器和生产过程又增加了功能，比如顺序、时间、计数和算术等，目前 PLC 已经广泛应用在复杂的自动化生产和控制行业中。

这种新型的工业控制装置以其简单易懂、操作方便、可靠性高、通用灵活、体积小、使用寿命长等一系列优点，很快在美国其他工业领域推广应用，不久便成功地应用于食品、饮料、冶金、造纸等工业。这一新型工业控制装置的出现，也受到了世界其他国家的高度重视。1971 年，日本从美国引进了这项新技术，很快研制出了日本第一台 PLC。1973 年，西欧国家也研制出他们的第一台 PLC。我国从 1974 年开始研制，于 1977 年开始进入工业应用。

进入 20 世纪 80 年代以来，随着大规模和超大规模集成电路等微电子技术的迅猛发展，以 16 位和少数 32 位微处理器构成的微机化 PLC 得到了惊人的发展，使得 PLC 在设计、性能价格比以及应用方面都有了新的突破，不仅控制功能增强，功耗和体积减小，成本下降，可靠性提高，编程和故障检测更为灵活方便，而且远程 I/O 和通信网络、数据处理以及图像显示的发展，还包括方便的调试和测试工具、仿真工具等，已经使 PLC 普遍用于控制复杂的连续生产过程。目前，可编程序控制器已成为工厂自动化的三大支柱之一。

1987 年 2 月，国际电工委员会（IEC）对可编程序控制器作了如下的定义：可编程序控制器是一种数字运算操作的电子系统，专为在工业环境下的应用而设计。它采用一类可编程序的存储器，用于其内部存储程序、执行逻辑运算、顺序控制、定时、计数和算术操作等面向用户的指令，并通过数字式或模拟式输入/输出，控制各种类型的机械或生产过程。可编

程序控制器及其有关外围设备，都按易于与工业控制系统组成一个整体、易于扩充功能的原则设计。

### 1.1.2 PLC 的主要特点

PLC 之所以应用如此广泛，与它优越的性能是分不开的。下面是其主要特点：

**1. 可靠性高、抗干扰能力强**

在传统的继电器控制系统中，使用了大量的中间继电器、时间继电器等。由于器件的老化、脱焊、触点的抖动、接触不良等现象，大大降低了系统的可靠性。而在 PLC 控制系统中，大量的开关动作是由无触点的半导体电路完成的，因触点接触不良等原因造成的故障大为减少。

另外，PLC 在硬件和软件方面还采取了以下强有力的措施，来提高其可靠性。

硬件方面，对所有的 I/O 接口电路均采用光电隔离，使工业现场的外电路与 PLC 内部电路之间电气上隔离；各模块均采用屏蔽措施，以防止辐射干扰；采用性能优良的开关电源并对供电系统和各输入电路均采用多种形式的滤波，以消除或抑制高频干扰；对所选用的器件进行严格的筛选；多采用模块式结构，一旦某一模块有故障，可以迅速更换模块，从而尽可能缩短系统的故障停机时间。

软件方面，PLC 具有良好的自诊断功能，一旦电源或其他软、硬件发生异常情况，CPU 立即采取有效措施，以防止故障扩大；PLC 设置了监视定时器（Watching Dog），如果程序循环的执行时间超过了规定值，则表明程序已进入了死循环，可以立即报警。

对于大型 PLC 系统，还可以采用由双 CPU 构成冗余系统或由三 CPU 构成表决系统，使系统的可靠性更进一步提高。

**2. 丰富的 I/O 接口模块**

PLC 针对不同的工业现场信号，如：交流或直流、开关量或模拟量、电压或电流、脉冲或电位、强电或弱电等信号，都能选择到相应的 I/O 模块与之匹配。对于工业现场的器件或设备，如：按钮、行程开关、接近开关、传感器及变送器、电磁线圈、控制阀等设备都能选择到相应的 I/O 模块与之相连接。

另外为了提高 PLC 的操作性能，它还有多种人机对话的接口模块；为了组成工业控制网络，还配备了多种通信联网的接口模块等。

**3. 采用模块化结构**

为了适应各种工业控制需要，除了单元式的小型 PLC 以外，绝大多数 PLC 均采用模块化结构。PLC 的各个部件，包括 CPU、电源、I/O 等均采用模块化设计，由机架及电缆将各模块连接起来，系统的规模和功能可根据用户的需要自行组合。

**4. 编程简单易学，系统的设计、调试周期短**

PLC 的编程大多采用类似于继电器控制线路的梯形图形式，对使用者来说，不需要具备计算机的专门知识，因此很容易被一般工程技术人员所理解和掌握。另外，由于 PLC 是通过程序实现对系统的控制，所以设计人员可以在实验室里设计和修改程序，还可以在实验室方便地进行系统的模拟及运行调试，使现场的工作量大为减少。

**5. 安装简单，维修方便**

PLC 不需要专门的机房，可以在各种工业环境下直接运行。使用时只需将现场的各种设

备与 PLC 相应的 I/O 端相连接，即可投入运行。

PLC 的故障率很低，且具有完善的自诊断和显示功能。当 PLC 或外部的输入装置和执行机构发生故障时，可以根据 PLC 各模块上的运行和故障指示装置或编程软件提供的信息，方便地查出故障的原因。由于采用模块化结构，因此一旦某一模块发生故障，用户可以通过更换模块的方法，使系统迅速恢复运行。

总之，可编程序控制器是一台计算机，它是专为工业环境应用而设计制造的。其具有丰富的输入/输出接口，并且具有较强的驱动能力。但可编程序控制器产品并不针对某一具体工业应用，在实际应用时，其硬件需根据实际需要进行选用配置，其软件需根据控制要求进行设计编程。

### 1.1.3 PLC 的主要应用

PLC 的应用领域包括：专用机床、纺织机械、包装机械、通用机械工程应用、控制系统、机床、楼宇自动化、电器制造工业及相关产业等。如图 1-1 所示是汽车车体生产线，图中的汽车生产线日产量为：1000～1400 辆汽车，其中有 200 条传输线，1000 台焊接机器人，所用西门子 PLC 则多达 1600 套。如图 1-2 所示为利用多台 PLC 控制的立体仓库。

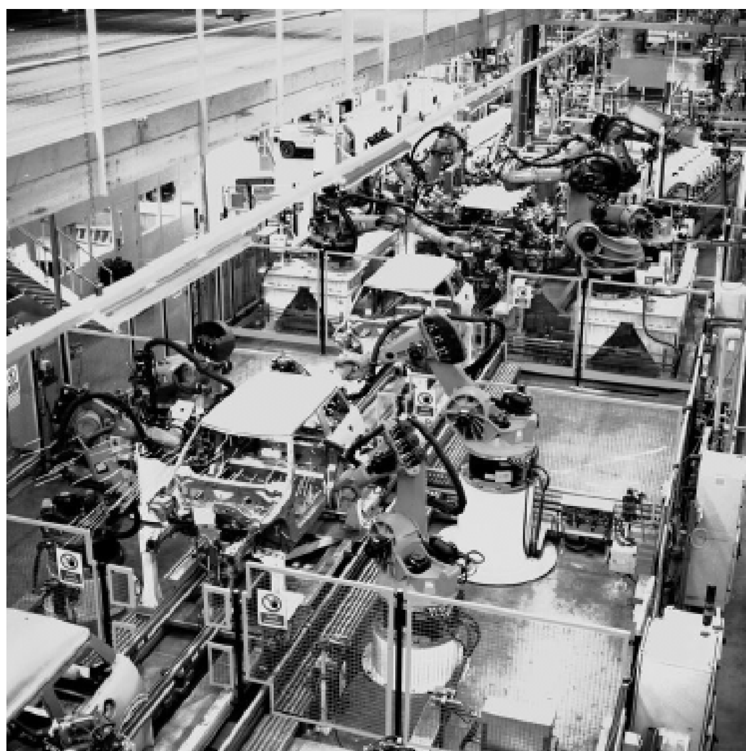


图 1-1　汽车车体生产线

总之，PLC 应用面广，功能强大，使用方便，已经成为当代工业自动化的主要支柱之一，在工业生产的所有领域得到了广泛的使用。



图 1-2　立体仓库

## 1.2　西门子 PLC 概述

### 1.2.1　西门子“全集成自动化”概念

“全集成自动化”（TIA）的概念最早源于欧洲，是在 1997 年德国法兰克福举办的 ACHMA 展会上推出西门子过程控制系统 SIMATIC PCS 7 时首次提出的，TIA 的推出当时被称为是一场技术革命，在工业行业中得到了高度的评价。经过几年的发展，TIA 不断地发展壮大，现在已经可以广泛地应用于工业生产中。全集成自动化思想就是用一种系统或者一个自动化平台完成原来由多种系统搭配起来才能完成的所有功能。应用这种解决方案，可以大大简化系统的结构、减少大量接口部件，可以克服上位机和工业控制器之间、连续控制和逻辑控制之间、集中与分散之间的界限。全集成自动化解决方案还可以为所有的自动化应用提供统一的技术环境，这主要包括：统一的数据管理、统一的通信、统一的组态和编程环境。

基于这种环境，各种各样不同的技术可以在一个用户接口下，集成在一个有全局数据库的总体系统中。工程技术人员可以在一个平台下对所有应用进行组态和编程。由于应用一个组态平台，工程变得简单，培训费用也大大降低，从而实现终极目的——帮助客户在投资阶段和整个系统服务周期里最大限度地降低成本。

西门子将工业以太网技术引入 TIA，在产品上集成以太网接口，使以太网进入现场级，从而实现元件自动化。而工业以太网正是业界广泛接受的通信标准，所以西门子的 TIA 是开放式架构的 TIA，其结构如图 1-3 所示。

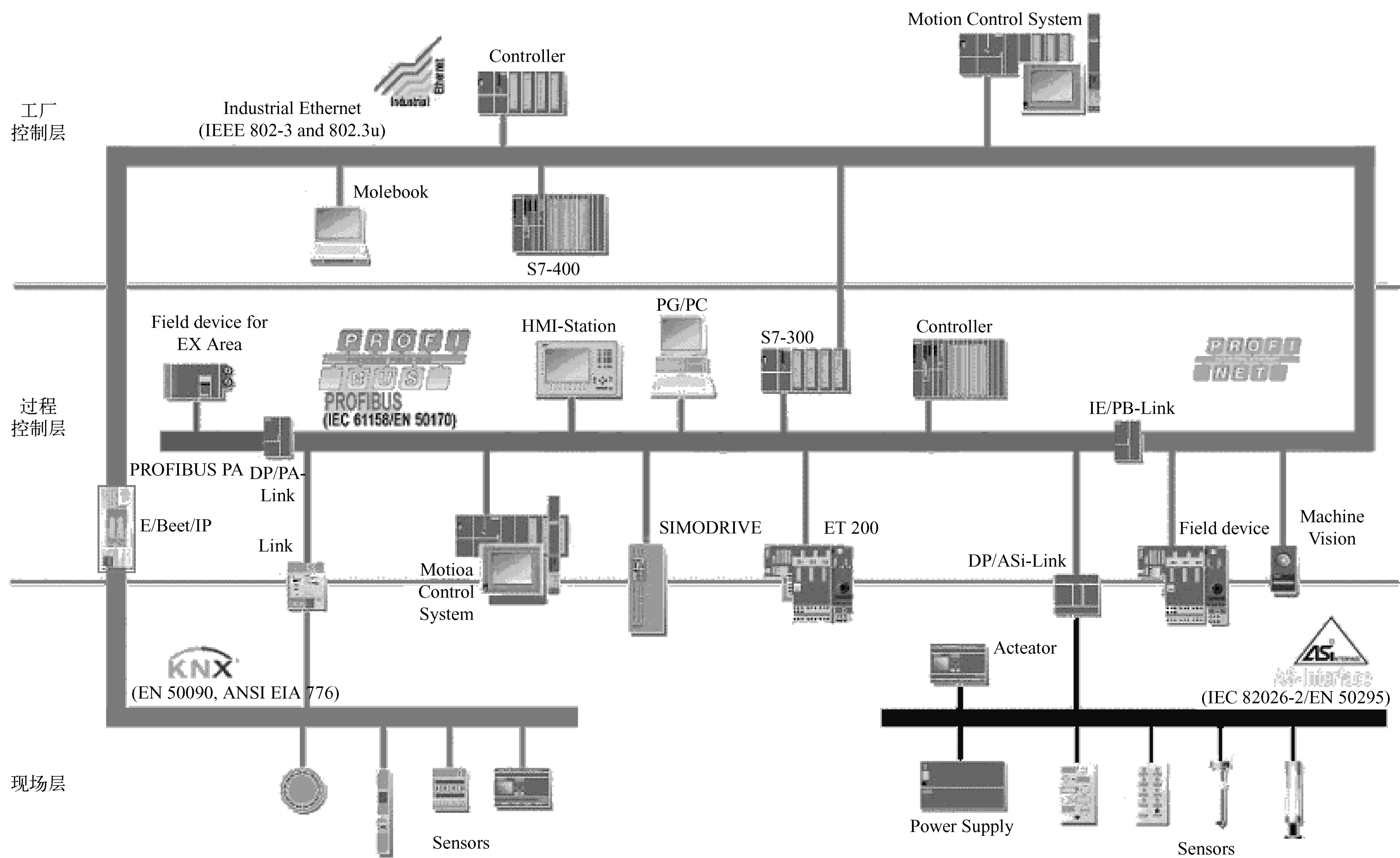

Motion Control System
Controller
Industrial Ethernet
(IEEE 802-3 and 802.3u)
工厂
控制层
Molebook
S7-400
Field device for
EX Area
PG/PC
HMI-Station
S7-300
Controller
PROFIBUS
(IEC 61158/EN 50170)
过程
控制层
IE/PB-Link
PROFIBUS PA
DP/PA-
Link
E/Beet/IP
Link
SIMODRIVE
ET 200
Field device
Machine
Vision
DP/ASi-Link
Motioa
Control
System
Acteator
KNX
(EN 50090, ANSI EIA 776)
(IEC 82026-2/EN 50295)
现场层
Sensors
Power Supply
Sensors

图 1–3　TIA 的结构

TIA 技术成功地实现了整个自动化系统在生产过程的水平和垂直方向上的集成，它基于网络化的“平台”理念，解决方案涵盖了生产制造的全过程。全集成自动化平台能够确保减少工程成本、提高生产效率、降低生命周期成本，这些都是当今世界制造业成本合理化不可或缺的要求。

### 1.2.2 西门子 PLC 产品

在全集成自动化概念中，PLC 是重要的组成部分。西门子公司是全世界最大的生产 PLC 产品的厂家之一。1975 年，西门子 S3 系列 PLC 正式进入自动化领域，该产品具有简单的二进制逻辑控制器和存储器。1979 年，S3 系列被 S5 系列取代，微处理器被广泛应用于 S5 系列产品中，使得该系列产品具有灵活的控制功能。20 世纪 80 年代初，S5 系列进一步升级为 U 系列，该系列产品在工业自动化领域得到了广泛的应用。

20 世纪 90 年代初，S7 系列 PLC 诞生，它具有体积小、速度快、标准化、具有网络通信功能、功能更强、可靠性更高、Windows 系统用户界面更良好等优势。其机型可分为 S7-200、S7-300 和 S7-400，2009 年又推出了 S7-1200。其中，S7-200 针对低性能要求的小型 PLC，根据 PLC 具体型号的不同，扩展性不同，最多可扩展 7 个模块。S7-300 是中小型 PLC，最多可扩展 32 个模块。S7-400 是用于中高级性能要求的大型 PLC，可以扩展 300 多个模块。S7-1200 不仅弥补了 S7-200 系列产品的不足，而且能与 S7-300 进行良好的链接，进一步满足中小型自动化系统的需求。

### 1.2.3 S7-300 系列 PLC

S7-300 是模块化中小型 PLC，既可以用于项目，也可以用于 OEM，通用性极强。在整个 TIA 系统销售中首屈一指，其应用的广泛性可见一斑。S7-300 是以在导轨上安装各种模块的形式组成系统。其品种繁多的 CPU 模块、信号模块和功能模块等几乎能满足各种领域的自动化控制任务。用户可以根据应用系统的具体情况选择适合的模块，维修时更换模块也很方便。信号模块和通信处理模块可以不受限制地插到导轨上任何一个槽，系统自行分配各个模块的地址。简单实用的分布式结构和强大的通信能力，使其应用十分灵活。S7-300 适用于通用领域，高电磁兼容性和强抗振动、冲击性，使其具有最高的工业环境适应性。

SIMATIC S7-300 具有多种不同的通信接口：

① 多点接口（MPI）集成在 CPU 中，用于连接编程设备。

② DP 接口用于连接 PC、人机界面系统及其他 SIMATIC S7/M7/C7 等自动化控制系统。

③ 多种通信处理模块用来连接 AS－i 接口、工业以太网和 PROFIBUS 总线系统。

④ 串行通信处理模块用来连接点对点的通信系统。

S7-300 的大量功能能够支持和帮助用户更简捷的编程，更好地完成自动化控制任务。主要功能如下：

① 高速的指令处理。0.6～0.1μs 的指令处理时间在中等到较低的性能要求范围内开辟了全新的应用领域。

② 浮点数运算。用此功能可以有效地实现更为复杂的算术运算。

③ 方便用户的参数赋值。一个带标准用户接口的软件工具给所有模块进行参数赋值，这样就节省了入门和培训的费用。

④ 人机界面（HMI）。方便的人机界面服务已经集成在 S7-300 操作系统内。因此人机对话的编程要求大大减少。SIMATIC 人机界面从 S7-300 中取得数据，S7-300 按用户指定的刷新速度传送这些数据。S7-300 操作系统自动地处理数据的传送。

⑤ 诊断功能：CPU 的智能化的诊断系统连续监控系统的功能是否正常、记录错误和特殊系统事件（例如超时模块更换等）。

⑥ 口令保护：多级口令保护可以使用户高度、有效地保护其技术机密，防止未经允许的复制和修改。

### 1.2.4　S7-400 系列 PLC

S7-400 是具有中高档性能的 PLC，适用于对可靠性要求极高的大型复杂控制系统。S7-400 也采用模块化结构，与 S7-300 系列 PLC 相比，模块的体积都比 S7-300 模块更大，尤其表现在高度上，所以每个 SM 模块的点数就更多。除了具有 S7-300 系统的功能外，S7-400 还具有如下的增强功能：

① 冗余设计的容错自动化系统。西门子系列的高可靠性 0.1μs 的指令处理时间在中等到较低的性能要求范围内开辟了全新的应用领域。

② 多 CPU 处理。在 S7-400 中央机架上，最多 4 个有多 CPU 处理能力的 CPU 同时运行。这些 CPU 自动地、同步地变换其运行模式，可以同步执行控制任务。使用多 CPU 中断（OB60）可以在相应的 CPU 中同步地响应一个事件。而且由于工作方式的复杂，CPU 模板上的指示灯也很多。

③ 扩展能力。中央机架只能插入最多 6 块发送型的接口模块，每个模块有两个接口，每个接口可以连接 4 个扩展机架，最多能连接 21 个扩展机架。扩展机架中的接口模块只能安装在最右边的槽。

④ 诊断功能。比 S7-300 强大，比如硬件中断功能最多提供多达 8 个，其他中断也比 S7-300 更多。

### 1.2.5　S7-1200 系列 PLC

西门子把最新的通信和控制技术应用在 S7-1200 产品上，瞄准的是中低端小型 PLC 产品线。西门子已经停止除在中国的 S7-200CN 系列以外的 S7-200 生产线，S7-200CN 以其低廉的价格还要争夺第三发展中国家的自动化市场份额，而在欧美低端市场将全部被 S7-1200 产品覆盖。S7-1200 硬件结构由紧凑模块化结构组成，系统 IO 点数、内存容量，均比 S7-200 多出 30%，充分满足了市场的针对小型 PLC 的需求。

S7-1200 的产品新特性如下：

① 紧凑模块化结构。S7-1200 产品延续了 S7－200 紧凑式结构，CPU1212C 和 CPU1211C 的宽度也仅有 90mm。通信模块和信号模块的体积也十分小巧，使得这个紧凑的模块化系统大大节省了空间，从而在安装过程中为用户提供了最高的效率和灵活性。

② 强大的控制功能。系统集成了 16 路 PID 的控制回路，并且 PID 都是能够支持自适应的快速功能块，并且提供了 PID 参数调试和观测的控制画面。系统集成了多达 6 个高速计数器（3 个 100kHz，3 个 30kHz），用于精确监视增量编码器、频率计数或对过程事件进行高速计数。系统集成了 2 个高速输出，可用作高速脉冲输出或脉宽调制输出。

③ 经典的编程模式。S7-1200 使用 SIMATIC STEP 7 Basic 工具编程，而这款的工具的使用风格基本与 STEP 7 一样，提供 LAD 和 FBD 两种编程语言并采用 OB 组织块、FB 功能块、FC 功能函数、DB 数据块的编程形式。

④ 复杂的数据结构。复杂的数据结构就是数组、结构等这样的多元素组成的数据单位。S7-1200 这款产品继承了 S7－300/400 中高端 PLC 所具备的数据结构，开始支持数组和结构等。

⑤ 指令参数的多态性。在西门子的经典的编程指令当中都是采用数据类型一致分类，例如加/减/乘/除的指令根据不同的数据类型是不同的指令，而在对 S7-1200 编程时不分数据类型，只是调用功能，当功能块放置在 Network 中时才会让用户选择是哪种数据类型，这就轻松地实现了参数的多态性。

⑥ 基于控制对象编程。基于控制对象的编辑和编程，添加控制对象只需要单击一下鼠标。添加新的对象时，工程组态系统的“添加新对象”窗口中会显示相关设置。根据对象的功能为对象命名。微调各种对象时，用户可以使用功能描述，分配完对象的所有信息后，编辑器中会立即打开该对象。

⑦ 集成 HMI 工程组态。SIMATIC STEP 7 Basic 包括功能强大的 HMI 软件 SIMATIC WinCC Basic，用于对 SIMATIC HMI 精简系列面板进行高效的编程和组态。HMI 是整个项目的一部分，HMI 数据可始终保持一致性。

⑧ 通信集成 Profinet 接口。S7-1200 支持传统的以太网 S7 通信，同样也支持 Profinet 工业以太网总线通信。

⑨ 灵活的第三方通信。S7-1200 配备了 CM 模块，支持 RS－232/485 以及自身以太网口通信。针对串行通信，RS－232/485 采用功能块配置帧通信的方式来完成数据流的通信，并且 S7-1200 支持 SEND_PTP 和 RCV_PTP 功能块串行通信的封装。

## 1.3 PLC 的组成

### 1.3.1 PLC 的基本结构

PLC 由中央处理单元（CPU）、存储器、输入单元、输出单元、通信单元、电源、扩展端口等单元有机结合而成。根据结构形式的不同，PLC 可分为整体式和组合式（也称模块式）两类。

整体式结构的 PLC 是将组成 PLC 的多个单元集成到一个箱体内构成主机，需要时再配合一些独立的 I/O 扩展单元组合使用。它的体积小，结构紧凑。小型机常采用这种结构，比如德国西门子（SIEMENS）公司的 S7-200 系列的 PLC。

组合式 PLC 是将组成 PLC 的多个单元分别做成相应的模块，各模块可以插在底板或导轨上，通过总线相互联系。组合式 PLC 系统配置灵活，中、大型 PLC 常采用这种方式，比如德国 SIEMENS 公司的 S7-300/400 系列的 PLC，美国 GE 公司的 VersaMax 系列的 PLC，法国施奈德公司的 Modicon Premium 系列的 PLC 等。

### 1.3.2 S7-300/400 系列 PLC 的组成

S7-300 系列 PLC 是模块化结构设计的 PLC，各个单独模块之间可进行广泛组合和扩展。

PLC 逻辑结构图如图 1-4 所示，以 CPU 为核心，通过总线扩展输入/输出模块、通信模块和闭环控制模块等其他模块组成系统。S7-400 系列 PLC 与 S7-300 相似。S7-300 系列 PLC 物理系统组成示意图如图 1-5 所示，主要由以下几部分组成：

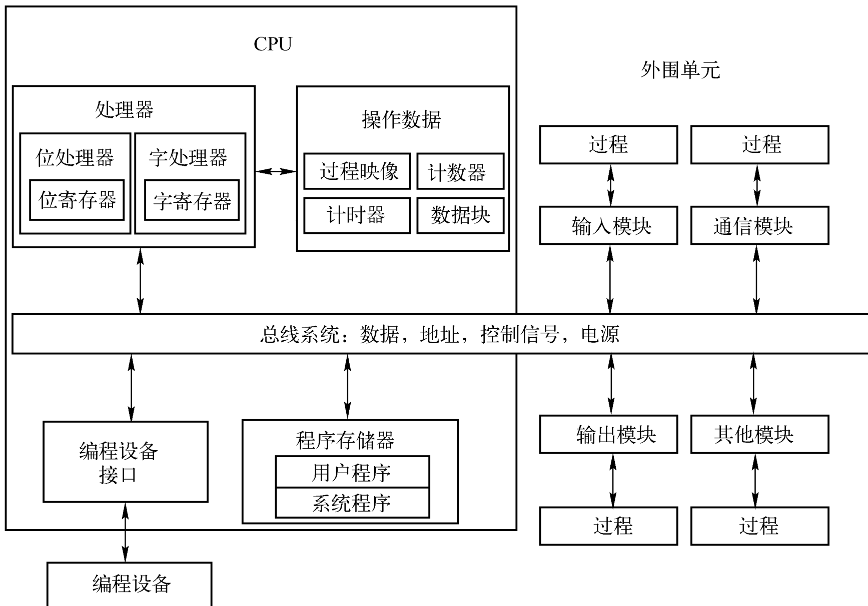



图 1-4　PLC 逻辑结构图

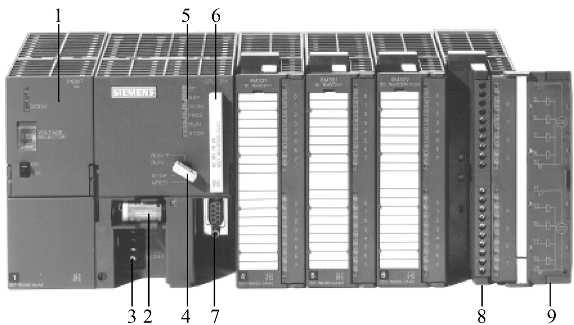



图 1-5　PLC 物理系统组成示意图

1—负载电源　2—后备电池（CPU 313 以上）　3—DC 24V 连接器　4—模式开关　5—状态和故障指示灯　6—存储器卡（CPU 313 以上）　7—MPI 多点接口　8—前连接器　9—前盖

（1）中央处理单元（CPU）

S7-300 系列 PLC 主要包括以下几种型号的 CPU：CPU312、CPU312C、CPU313C、CPU313C-PtP、CPU314C-2DP、CPU315C 等，每种 CPU 有其不同的性能，例如：型号中含有“C”的表示带有计数器；型号中含有“DP”的表示带有 9 针 DP 接口；型号中含有“PtP”的表示带有 15 针 PtP 接口；还有的 CPU 上集成有输入/输出点等。

（2）负载电源模块（PS）

负载电源模块用于将 120/230V 交流电源转换为 24V 直流电源，供其他模块使用。它与

CPU 模块和其他信号模块之间通过电缆相连，不是通过背板总线连接。额定输出电流有 2A、5A 和 10A 三种，可根据负载要求选择。

（3）信号模块（SM）

信号模块是数字量输入/输出和模拟量输入/输出模块的总称，它们使不同的过程信号电平和 PLC 的内部信号电平相匹配。信号模块主要有 SM321（数字量输入模块）、SM322（数字量输出模块）、SM331（模拟量输入模块）、SM332（模拟量输出模块）。每个模块都带有一个背板总线连接器，用于与 CPU 和其他模块间的数据通信；还可配一个螺紧型前连接器，通过它外部过程信号可方便地连接到信号模块。模拟量输入模块可以输入热电阻、热电偶、DC 4～20 mA 和 DC 0～10V 等多种不同类型和不同量程的模拟信号。

（4）通信处理模块（CP）

通信处理模块用于 PLC 之间、PLC 与计算机和其他智能设备之间的通信，可以将 PLC 接入 PROFIBUS－DP、AS－i 和工业以太网，或用于实现点对点连接等。它可以减轻 CPU 处理通信的负担，并减少用户对通信功能的编程工作。

（5）功能模块（FM）

功能模块主要用于对实时性和存储容量要求高的控制任务，例如计数器模块 FM350 可完成高速计数功能；伺服电动机定位模块 FM354 可完成定位操作（开环或闭环定位）；闭环控制模块 FM355 可完成闭环控制功能。

（6）接口模块（IM）

接口模块用于多机架配置时连接主机架（CR）和扩展机架（ER）。S7-300 通过分布式的主机架和连接的扩展机架（最多可连接 3 个扩展机架），可以操作多达 32 个模块。

## 1.4 PLC 的工作原理

### 1.4.1 工作原理

本节以电动机正、反转的实例来阐述 PLC 的工作原理。如图 1－6 所示，输入 I0.0、I0.1 和 I0.2 分别采集电动机停止、正转和反转的按钮信号，输出 Q0.0 和 Q0.1 控制电动机的正转和反转。

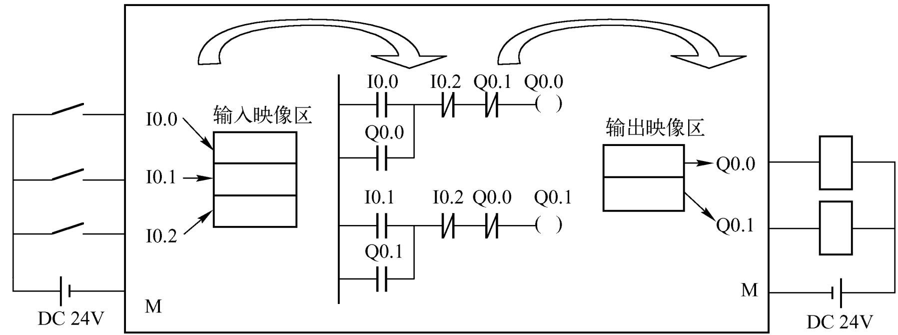



图 1-6　PLC 的外部接线和梯形图

系统上电或由 STOP 模式切换到 RUN 模式时，CPU 要执行一次复位操作，包含如下两个操作步骤：

1）清除没有保持功能的位存储器状态、定时器和计数器状态，清除中断堆栈和块堆栈的内容等。

2）执行系统起动组织块 OB100。如果用户想使系统在上电后做一些初始化操作，就可以在 OB100 中编写程序，否则用户完全可以忽略这个组织块。需要注意的是：OB100 只在复位后被执行一次。

整个 PLC 的工作过程是以循环扫描的方式进行的，重复执行一个循环工作周期。以下四个步骤就是 PLC 程序执行的一个循环工作周期：

1）操作系统起动循环时间监控。

2）CPU 将输出映像区中的数据写到输出模块。

3）CPU 读取输入电路的接通/断开状态并存入输入映像区。

4）CPU 处理用户程序，执行用户程序中的指令，并实时更新内存映像区。

在第 1 阶段，操作系统起动用户设置的监控循环时间。设置的方法参见第 8 章。

在第 2 阶段，CPU 将输出映像区中的数据状态传送到输出模块，用于控制与输出点连接的继电器线圈。若上次循环工作周期中输出映像区中 Q0.0 状态为“0”，而这次 Q0.0 得电状态为“1”时，控制电动机的继电器线圈通电，其常开触点吸合，电动机正转；反之，控制电动机的继电器线圈断电，其常开触点断开，电动机停止工作。

在第 3 阶段，PLC 通过输入模块采集外部电路的接通断开状态，并写入到输入映像区中。若外部电路开关 SB1 闭合时，对应的输入映像位 I0.0 状态为“1”，在梯形图中对应的 I0.0 常开触点闭合，常闭触点断开，反之亦然。

在第 4 阶段，在 CPU 执行程序指令时，从映像区特别是输入映像区中读出程序中所用元件的“0”、“1”状态，并执行指令，将运算结果实时写入到对应的映像区中。需要注意的是：在程序执行阶段，即使外部输入信号的状态发生了变化，输入映像区对应的元件位也不会随之立即改变，只能等到这个循环扫描周期结束，下个循环扫描周期开始时才能被更新。

在 S7-300 中，系统不断地调用组织块 OB1（相当于 C 语言中的主函数），在主函数中调用其他子程序，包括子程序（指逻辑块 FC 或 FB）和系统自带的子程序（指系统逻辑块 SFC 或 SFB 中）。

在实际工程应用中，中断是不可缺少的工作方式，循环工作过程可以被某些事件中断。S7-300 和 S7-400 的 CPU 为用户提供了多种中断方式，以下几种较为常用：

① 中断源通过外部电路的输入进入系统，中断服务程序需事先存入组织块 OB40。

② 系统提供了某些组织块为中断工作方式服务，有 OB10（日期时间中断组织块）和 OB20（延时中断组织块）两种。

总之，CPU 从第一条指令开始，逐条地执行用户程序，并且循环重复执行。执行指令时，从元件映像区中将有关编程元件的 0/1 状态读出来，并根据指令的要求执行相应的逻辑运算，实时更新映像区，最后的运算结果输出到生产过程的执行机构中。

### 1.4.2　循环时间和响应时间

循环时间是指一个程序执行一个周期所占有的时间，例如执行一个 OB1 的周期时间。一般 PLC 的循环时间在数十到一百多毫秒之间，每个用户程序的循环时间并不相同，主要受到以下情况的影响而延长：不同的程序路径；有过程中断的处理；有诊断和故障的处理；通信方式的不同等。



图 1-7　蒙娜丽莎

PLC 的响应时间是指从检测到一个输入信号至相应输出信号发生改变之间的时间。响应时间主要取决于循环时间和信号模块、集成 I/O 的输入和输出延迟等因素，一般也是毫秒级的。当购买 PLC 时，它的响应时间是我们不得不考虑的一个事实。就像人的大脑一样，PLC 也需要一定的时间去做出反映。例如：当你阅读这段文字时，你会看到一幅蒙娜丽莎画像（见图 1-7），在你的大脑意识到“哦，这儿有一幅画！”之前，你的眼睛事实上已经看到了这幅画。在这个例子中，你的眼睛可以被认为是传感器，它们被连接到大脑的输入电路中。大脑的输入电路需要花一定的时间（即输入响应时间）去反映你的眼睛所见（如果你已饮酒，那么这段时间将会更长）。然后，你的大脑意识到你已经看到了画并且处理这些数据（即程序处理时间）。最终，你的嘴得到大脑发出的信号，说出“看啊！蒙娜丽莎变丑了！”（即输出响应时间）。输入响应时间、程序处理时间和输出响应时间共同构成了 PLC 的响应时间。

## 习题

1. 简述 PLC 的定义。
2. PLC 有哪些主要特点？
3. PLC 可以应用在哪些领域？
4. 与继电器控制系统相比，PLC 控制系统有哪些优点？
5. 简述“全集成”概念。
6. S7-300 系列 PLC 的基本功能有哪些？
7. S7-300 系列 PLC 主要由哪几类模块构成？
8. 简述 PLC 的工作原理。

# 第 2 章　S7-300/400 结构体系

## 2.1　S7-300 的 CPU 模块

### 2.1.1　CPU 的分类

S7-300 PLC 具有 20 种不同型号的 CPU，按性能等级划分，可以涵盖各种应用范围。主要分为以下几类：

① 6 种紧凑型 CPU：CPU312C、CPU313C、CPU313C-2PtP、CPU313C-2DP、CPU314C-2PtP、CPU314-2DP。这些 CPU 的共同特点是带有集成的数字量输入和输出或兼有模拟量的输入和输出，CPU 运行时需要微存储器卡。多数 CPU 都适用于具有较高要求的系统。型号中带“PtP”的 CPU 除编程端口外还带有第 2 个串口；型号中带“2DP”的 CPU 带有 PROFIBUS DP 主站/从站接口。紧凑型 CPU 模块如图 2-1 所示，非紧凑型 CPU 模块如图 2-2 所示。

② 3 种重新定义的标准 CPU：CPU312、CPU314、CPU315-2DP。CPU312 适用于对处理速度中等要求的小规模应用。CPU314 和 CPU315-2DP 分别适用于对程序量中等要求和大规模要求的应用，两者对二进制和浮点数运算具有较高的处理性能。CPU315-2DP 还带有 PROFIBUS DP 主站/从站接口，而且可用于大规模的 I/O 配置。这类 CPU 运行时需要微存储器卡。

③ 5 种标准 CPU：CPU313、CPU 314、CPU 315、CPU 315-2DP、CPU 316-2DP。与第二类相似，分别用于中、大规模的应用。这类 CPU 运行时需要微存储器卡。

④ 故障安全型 CPU315F-2DP：不需要对故障 I/O 进行额外的接线，可以组态为一个故障安全型自动化系统，可满足安全运行的需要。模块集成有 PROFIBUS DP 主站/从站接口。这类 CPU 运行时需要微存储器卡。

⑤ 4 种 SIMATIC S7-300 户外型 CPU：CPU 312IFM、CPU 314 IFM、314 户外型和 315-2 DP。户外型 CPU 适用于恶劣的环境。312IFM 和 314 IFM 也是紧凑型 CPU，集成了数字量输入/输出，前者适用于小系统，后者适用于对响应时间和特殊功能有较高要求的系统。


图 2-1　紧凑型 CPU 模块


图 2-2　非紧凑型 CPU 模块

⑥ 高端 CPU 318-2 DP：具有大容量程序存储器以及 PROFIBUS-DP 主/从接口，可用于大规模的 I/O 配置，建立分布式 I/O 结构。

各种不同型号 CPU 的具体性能指标请参考光盘中的手册《SIMATIC S7-300 可编程控制器》。

### 2.1.2　CPU 的面板

S7-300 的 CPU 种类很多，具有不同的功能，所以其面板也不是完全相同。CPU 313C-2DP 的前面板示意图如图 2-3 所示，主要有如下的部件：

1）卡槽。可插入微存储卡 MMC，可用做装载存储器或便携式存储媒介。由于多数 CPU 都没有集成的装载存储器，因此必须在购买 CPU 的同时也配置 MMC，否则 CPU 将无法工作。插入存储器卡前，把 CPU 切换到 STOP 状态，或关断电源。

2）状态和故障指示灯。

3）CPU 的 3 位模式开关（在某些模块上是可旋转的 4 位钥匙开关，如图 2-3 所示）。表 2-1 说明了 CPU 的模式开关及含义。

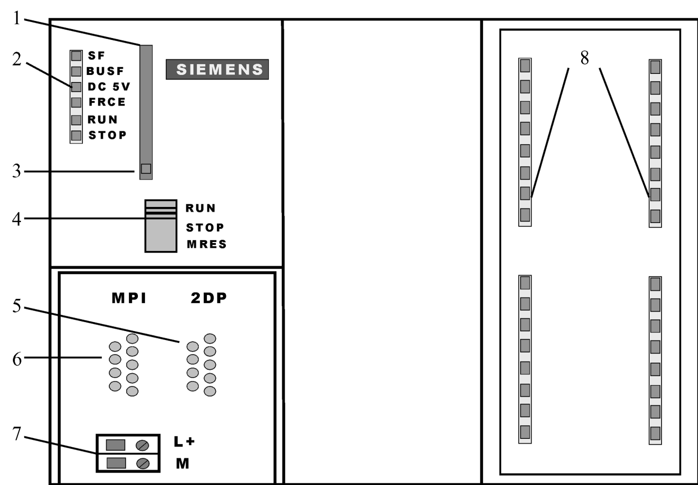



图 2-3　CPU 313C-2DP 的前面板示意图

1—卡槽　2—状态和故障指示灯　3— MMC 弹出器按钮　4—3 位模式开关　5—2 个 DP 接口
6—多点接口 MPI　7—电源接线端子 24V 电源的连接　8—集成 I/O

表 2-1　CPU 的模式开关及含义

| 位　置 | 含　义 | 说　　明 |
| --- | --- | --- |
| RUN-P | 运行和编程模式 | CPU 不仅执行用户程序，还可以修改用户程序 |
| RUN | 运行模式 | CPU 正在执行用户程序 |
| STOP | 停止模式 | CPU 不执行用户程序 |
| MRES | 存储器复位 | 可使 CPU 存储器复位 |

有如下情况执行 CPU 存储器复位：

① 当第一次起动前。

② 当新的完整的用户程序下载前。

③ 如果 CPU 要求存储器复位时（STOP LED 闪烁）。

用模式开关执行 CPU 存储器复位的操作步骤如下：

① 合上电源开关。

② 把开关转到 STOP 位置。

③ 把开关转到 MRES 位置（存储器复位），并保持在这个位置直到 STOP 指示灯再次变亮（大约 3s）。

④ 把钥匙开关转回 STOP 位置，然后再转到 MRES，直到 STOP 指示灯再次亮 1s。

4）MMC 弹出器按钮。轻轻一按，MMC 卡便自动弹出。

5）2 个 DP 接口。用于接入网络或与上位机连接。

6）多点接口 MPI。连接到编程设备或另一台可编程序控制器的多点接口（MPI）。

7）电源接线端子 24V 电源的连接。用于与电源模块的供电连接。

8）集成 I/O。集成了 16 个数字量输入点和 16 个数字量输出点。

### 2.1.3　CPU 的存储器

存储器用于存储系统程序和用户程序（包括组态信息），主要类型有 RAM（随机存取存储器）、ROM（只读存储器）、Flash EPROM（快闪存储器）和 EEPROM（电可擦除的只读存储器）等物理存储器。其性能各不相同：RAM 存储用户的程序，ROM 用来存储 PLC 的系统程序，而 Flash EPROM 和 EEPROM 兼有 ROM 非易失性和 RAM 的随机存取的优点，用来存放用户程序和需要长期保存的重要数据。

CPU 存储器从逻辑上可以分为三个区域：

（1）装载存储器

装载存储器位于微存储卡 MMC 中，其容量与 MMC 的容量一致，用来保存项目的没有符号表和注释的用户程序以及硬件组态的数据。

（2）工作存储器（RAM）

工作存储器用来存储程序处理所需的那一部分用户程序及数据。RAM 集成在 CPU 中，不能被扩展。复位 CPU 中的存储器，存储在 RAM 中的程序将丢失，但存储在 MMC 中的程序却安然无恙。

（3）系统存储区

系统存储区的物理介质是 RAM，也集成在 CPU 中，不能被扩展，包括：标志位、定时器和计数器的地址区；I/O 的过程映像；局域数据等。

## 2.2　S7-300 的信号模块

输入/输出模块统称为信号模块（SM）。下面以 S7-300 为例，介绍 S7-300 系列 PLC 的信号模块。S7-300 的数字 I/O 模块包括数字输入模块 SM321 和数字输出模块 SM322 等，可用于连接数字传感器和执行元件数字 I/O，使 PLC 灵活地与任务相适应。模拟 I/O 模块包括模拟输入模块 SM331 和模拟输出模块 SM332 等，通过这些模块可将模拟传感器和执行元件与 S7-300 相连。此外还有用于调试的 SM374，它是 16 点数字量模块，通过旋动面板上的开

关，可实现自由选择三种输入/输出点数：输入 16 点；输出 16 点；输入/输出各 8 点。

SIMATIC S7-300 的数字输入/输出模块用于连接开关、2 线接近开关（BERO）、电磁阀、接触器、小功率电动机、灯和电动机起动器等，将控制过程的外部数字量电平转化为S7-300 的内部信号电平，并将 S7-300 内部信号电平转化为控制过程所需的外部信号电平。

模拟 I/O 模块具有下列优点：

① 优化配合。模块可任意组合以配合所需输入/输出点数量，没有必要增加投资。

② 强大的模拟技术。不同的 I/O 范围和高分辨率允许与众多不同的模拟传感器和执行元件相连。

信号模块结构紧凑，组装简单，接线方便。该类模块安装在 DIN 标准导轨上并通过总线连接器与相邻模块相连接。

### 2.2.1 数字量模块

#### 1. 数字量输入模块 SM321

数字量输入模块用于采集现场过程的数字信号电平（直流信号或交流信号都可），并把它转换为 PLC 内部的信号电平。一般数字量输入模块连接外部的机械触点和电子数字式传感器。数字量模块的输入、输出电缆最大长度为 1000 m（屏蔽电缆）或 600 m（非屏蔽电缆）。

用于采集直流信号的模块称为直流输入模块，其名称含有 VDC，额定输入电压为直流 24V；用于采集交流信号的模块称为交流输入模块，其名称含有 VAC，额定输入电压为交流 120V 或 230V。如果信号线不是很长，PLC 所处的物理环境较好，电磁干扰较轻，应考虑优先选用以 DC 24V 的直流输入模块。交流输入方式适合于有油雾、粉尘的恶劣环境下使用。

对于用户来说，数字量输入模块 SM321 有四种型号的模块可供选择，分别是 DC16 点输入、DC32 点输入、AC16 点输入、AC8 点输入模块。

在把数字量输入量模块与电源和 CPU 模块安装在导轨上后，需要连接电缆线为 SM321 供电。

模块上的每个输入点的输入状态是用一个绿色的发光二极管来显示的，输入开关闭合即有输入电压时，二极管点亮。

#### 2. 数字量输出模块 SM322

数字量输出模块将 PLC 内部信号电平转换成外部过程所需的信号电平，同时具有隔离和功率放大的作用。模块能连接继电器、电磁阀、接触器、小功率电动机、指示灯和电动机软起动器等负载。

按负载回路使用的电源不同，数字输出模块可以分为直流输出模块、交流输出模块和交直流两用输出模块。按输出开关器件的种类不同，它又可分为晶体管输出方式、晶闸管输出方式和继电器触点输出方式。

以上两种分类方式又有密不可分的关系。晶体管输出方式的模块只能带直流负载，属于直流输出模块；晶闸管输出方式的模块属于交流输出模块；继电器触点输出方式的模块属于交直流两用输出模块。从响应速度上看，晶体管响应最快，继电器响应最慢；从安全隔离效果及应用灵活性角度看，继电器输出型的性能是最好的。

一般情况下，用户多采用继电器型的数字输出模块，而它的价格也相对高些。继电器输

出模块的额定负载电压范围较宽，输出直流最小是 DC 24V，最大可到 DC 120V；输出交流的范围是 AC 48 ~230V。

SM322 DO 32 ×24 VDC/0. 5A（订货号为6ES7 322 -1BL00 -0AA0）有 32 个直流输出点，8 点为一组，具有组隔离功能，适用于电磁阀、直流接触器和指示灯。

SM322 DO 8 ×Rel. 230 VAC/5A（订货号为6ES7 322 -1HF10 -0AA0）是 8 个输出点的模块，负载电压 DC 24 ~120 V 或 AC 24 ~230V，1 点为一组，也具有组隔离功能。在使用前，必须先连接好模块的工作电源 AC24V。若使用交流输出方式，每个交流输出都可用三个端子，其中一个接外电路的交流接触器，另两个端子都可接外部交流电源，因为这两个端子的内电路已串联，所以只需将其中一个接入交流电源即可。

还有一类模块兼有输入和输出两种功能，例如 SM 323 就是数字量输入/输出模块，额定的输入电压和负载电压都是 DC 24V，不能用于交流使用。SIEMENS 提供了两种型号的 SM 323，一种是 DI 16/DO 16 ×24 VDC/0. 5A（订货号为6ES7 323 -1BL00 -0AA0），具有 16 点输入（16 点为 1 组）和 16 点输出（8 点为 1 组）；另一种是 DI 8/DO 8 ×24 VDC/0. 5A（订货号为6ES7 323 -1BHx1 -0AA0），具有 8 点输入和 8 点输出（8 点为 1 组），都具有组隔离功能。

3. 数字量仿真模块 SM374

仿真模块 SM374 常用于调试程序和实验中，直接用模块上的开关来模拟数字量输入/输出信号，同时用 LED 灯显示输入/输出的状态。模块上有 16 个 I/O 点，可以工作在三种模式下：16 点输入、16 点输出和 8 点输入/8 点输出。工作模式的选择通过用螺钉旋具旋转模块上功能设置开关的位置实现。

需要注意的是，在 STEP 7 软件的硬件组态中，没有直接给出模块 SM374 的型号和订货号，所以用户需要根据模块的工作模式选择它的组态。例如：将 SM 374 设置为 8 点输入和 8 点输出，则选择一个 8 点输入/8 点输出的数字量模板的订货号如 6ES7 323 -1BH00 -0AA0 即可。

### 2. 2. 2 模拟量模块

在生产过程中，存在大量的物理量，如压力、温度、速度、旋转速度、pH 值、粘度等，而 PLC 作为数字控制器不能够直接处理连续数据。因此，必须对这些物理量进行离散化，这就是模/数（A/D）转换。另外，控制这些物理量的执行器多数接收的也是模拟量，所以，PLC 处理过的数据还必须进行数/模（D/A）转换成连续的模拟信号（如电压、电流）来驱动执行器动作，从而达到控制物理量的目的。

S7-300 系列的模拟量模块主要有 SM331、SM332 和 SM334。模拟量模块 SM331 只具有模拟量输入通道，用于连接电压和电流传感器、热电耦、电阻器和电阻式温度计，将扩展过程中的模拟信号转化为 S7-300 内部处理用的数字信号。模拟量模块 SM332 只带有模拟量输出通道，用于将 S7-300 与执行元件相连，将数字输出值转换为模拟信号。模拟量输入/输出模块 SM334 兼有模拟输入和模拟输出的功能。SM331 与 SM332 模块背面还有量程盒，用户在使用的时候应该根据实际情况选择传感器的类型、量程，以及能够正确地接线。而 SM334 不需要设置量程盒，该模块使用起来比较简单。

1. 模拟量输入模块 SM331

SM331 有多路输入通道可供选择的模块，如 AI2 ×12bit、AI4 ×12bit、AI8 ×12bit、AI8 ×16bit、AI8 ×RTD 等。下面以 AI4 ×12bit 模块为例来说明 SM331 的使用。

SM331（AI8 ×12bit）是一个具有 8 个输入测量通道的模块，2 个通道为一组。通过改变模块背面的量程盒的位置，每个通道测量可以分别设置 4 个不同的测量范围，如图 2-4 所示。

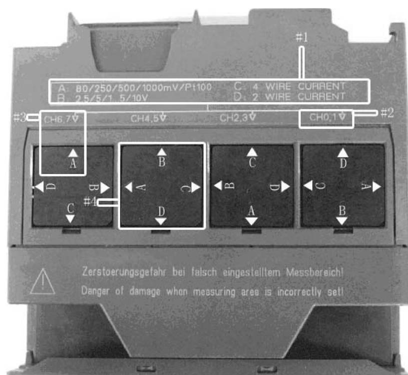


图 2-4　SM331 的量程盒设定

1 号框内给出了 4 种测量信号的范围，分别如下：

A：+/-80/250/500/1000 mV/Pt100 的热电偶、热电阻测量。

B：+/-2.5V/5V/1-5V/10V 电压测量。

C：标准 4~20 mA、0~20 mA、+/-10 mA、+/-3.2mA、+/-20mA 的 4 线制电流（4DMU）。

D：标准 4~20mA 的 2 线制电流（2DMU）。

2 号框内给出了 1 号通道组，包括两个通道 CH0 和 CH1；2 号通道组，包括通道 CH2 和 CH3；3 号通道组，包括通道 CH4 和 CH5；4 号通道组，包括通道 CH6 和 CH7。

4 号框显示的是一个量程盒，每个通道组内的两个通道的测量信号可以设定成一号框内指示的 A、B、C 和 D 4 种信号模式。

在 3 号框内，当 A 的箭头指向通道号的时候，说明 CH6 和 CH7 的输入信号为 +/-80/250/ 500/1000mV/Pt100 的电压信号。类似地，其他通道组的量程盒可以根据实际情况选择恰当的信号类型，如 3 号框，只要将该信号的类型号（A、B、C 和 D）的箭头指向通道号即可。

如果要改变某个通道组的测量范围，采取以下步骤调整量程卡：

1）使用螺钉旋具将量程卡从模拟量输入模块中松开，如图 2-5 所示。

2）将量程卡（正确定位 1）插入模拟量输入模块中。所选测量范围为指向模块上标记点 2 的测量范围，如图 2-6 所示。

模拟量模块发生故障时，总是以模块上的 SF 指示灯来指示。而且，故障信息也可以从诊断缓冲区中获得。

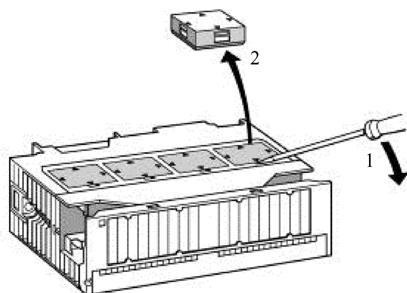


图 2-5　将量程卡从模拟量输入模块中松开

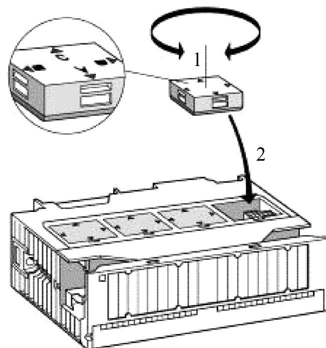


图 2-6　将量程卡插入模拟量输入模块

2. 模拟量模块 SM334

SM334 既有模拟量输入通道，也有输出通道。该模块本身没有量程盒选择。SM334 模拟量模块有不同的订货号，每个订货号的产品在功能上略有差别。以订货号为 6ES7 334－0CE01－0AA0 的 SM334 的模块为例，该模块有 4 个输入通道、2 个输出通道。每个输入通道既可以连接电压信号，也可以连接电流信号，但不能在连接电压信号的同时又连接电流信号。每个输出通道也是既可以连接电压信号，也可以连接电流信号，但也不能同时使用。

## 2.3 S7-300 的特殊模块

### 2.3.1 通信处理模块 CP 34x

通信处理模块为 PLC 应用系统接入 PROFIBUS 网络、工业以太网和 AS－i 等网络提供了极大的便利。通过集成在 STEP 7 中的参数化工具可进行简便的参数设置。

CP 340 通信处理器是串行通信中最经济的模块。

CP 341 通过点到点连接，用于高速、强大的串行数据交换，以减轻 CPU 的负担。

CP 343－2 是连接 S7-300 和 ET 200M 的 AS－i 主站。通过连接 AS－i 接口，每个 CP 最多可访问 248 个数字量输入和 186 个数字量输出。通过内部集成的模拟量值处理程序，可以对模拟量值进行处理。

CP 342－5 是连接 S7-300 和 C7 到 PROFIBUS－DP 总线系统的低成本的模块。它减少 CPU 的通信任务，同时支持其他的通信连接。

CP 343－1 在工业以太网上独立处理数据通信。该模块有其自身的处理器，符合国际标准的 1～4 层协议。

### 2.3.2 计数器模块 FM 350 和 CM 35

模块的计数器均为 0～32 位或 31 位加减计数器，可以判断脉冲的方向，模块给编码器供电。其具备比较功能，达到比较值时，通过集成的数字量输出响应信号，或通过背板总线向 CPU 发出中断。通过集成的数字量输入直接接收起动、停止计数器等数字量信号。

FM 350－1 是智能化的单通道计数器模块，广泛用于单纯的计数任务。该模块依据可直接连接的门信号检测最高达 500kHz 的增量编码器脉冲。它有三种工作模式：连续计数、单向计数和循环计数。模块有 3 个数字量输入（1 个用于门起始、1 个用于门结束、1 个用来设定计数器），2 个数字量输出。

FM 350－2 是用于计数和测量任务的智能型 8 通道计数器模块。该模块提供 7 种不同的工作方式，以便与希望的应用快速而简单地相配合，分别为连续计数、单向计数、循环计数、频率测量、速度测量、周期测量和比例运算。

CM 35 是 8 通道智能计数器模块，可广泛用于计数及测量任务，也可用于最多 4 轴的简单定位任务。它有 8 个计数输入端，可选 5V 或 24V 电平；8 个数字输出点用于对模块的高速响应输出，也可由用户程序指定输出功能。CM35 有四种工作方式：脉冲计数器（8 通道）、定时器（8 通道）、周期测量（8 通道）和简易定位（4 轴）。

### 2.3.3 位置控制与位置检测模块 FM 35x

FM 351 是双通道定位模块，用于快进给和慢速驱动的定位，可以控制两个相互独立的轴的定位。该模块最好通过由接触器或变频器控制的标准电动机来调整轴或设定轴定位。

FM 352 电子凸轮控制器是非常高速的控制器。通过一个传感器检测轴的位置，然后通过集成的输出端触发控制指令。即使在低端应用范围，FM352 也是机械式凸轮控制器的低成本替代。凸轮轨迹的 13 个数字输出端，用于迅速将控制信号传送到过程中。每个凸轮基于速度的动态移位，用于对连接的执行器进行滞后补偿。被控单元能直接与模块相连接。中间继电器只有在执行器有较高的功率消耗时才需要采用。

FM 352－5 高速布尔处理器可以非常快速地进行布尔控制（循环周期 1μs）。该模块集成 8 点数字量标准输入和 8 点数字量输出，当用于 DC24V 传感器输入作为数字量时最多有 12 个数字量输入点。指令集包括：位指令、定时器、计数器、分频器、频率发生器、移位寄存器。1 个通道用于连接 1 个 24V 增量编码器、1 个 5V 编码器（RS 422）或 1 个串口绝对值编码器。

FM 353 步进电动机定位模块是通过步进电动机实现各种定位任务的智能模块。该模块可以用于任何简单的点到点定位，也可以用于对响应、精度和速度有极高要求的复杂运动模式，是高速机械设备定位任务的理想解决方案。它将脉冲传送到步进电动机的功率驱动器，由脉冲数量决定移动路径的长度，同时，脉冲频率控制移动速度。FM 353 集成了 4 点数字量输入和 4 点数字量输出。步进电动机的 FM 353 定位模块可用于定位如：进给轴、调整轴、设定轴和传送带式轴（直线和旋转轴）。

FM 354 伺服电动机定位模块是通过伺服电动机实现广泛的定位任务的智能模块，可用于简单的点对点定位任务，也可用于对响应、精度和速度要求极高的复杂运动方式，是高速机械设备定位任务的理想解决方案。FM 354 可用于下列应用的伺服定位：进给轴、调整轴、设定轴、传输轴（直线和旋转轴）和传输带。FM 354 集成了 4 点数字量输入和 4 点数字量输出。它用模拟驱动接口（－10～＋10V）控制驱动器，通过编码器（SSI 或增量编码器）检测的目前轴的位置，来修正输出电压。

### 2.3.4 闭环控制模块 FM 355

FM 355 闭环控制模块是用于闭环控制任务的智能化 4 通道模块。它可用于温度控制、压力控制、流量控制、液面控制等。FM 355 有两种不同的形式：FM 355C 作为连续控制器，带有 4 个模拟输出端用于控制模拟执行元件；FM 355S 作为步进或脉冲控制器，带有 8 个数字输出端用于控制电动机（集成）驱动的执行元件或二进制控制的执行元件。FM 355 提供一个组态软件包，通过参数化窗口可以对参数进行初始化和起动。

### 2.3.5 称重模块 SIWAREX

SIWAREX U 是紧凑型电子秤，用于化学工业和食品工业等行业测定料仓和斗的料位，对起重机载荷进行监控，对传送带载荷进行测量或对工业提升机、轧机超载进行安全防护等。该模块有下列功能：衡器的校准、重量值的数字滤波、重量测定、衡器置零、极限值监控和模块的功能监视。

SIWAREX M 是有校验能力的电子称重和配料单元。可以用它组成多料秤称重系统，以准确无误地关闭配料阀，达到最佳的配料精度。该模块有下列功能：置零和称皮重、自动零点追踪、设置极限值（Min/Max/空值/过满）、操纵配料阀（粗/精配料）、称重静止报告和配料误差监视。

## 2.4 硬件模块的安装

S7-300 可供选择的模块不仅种类较齐全，而且配有标准的附件。认识并如何利用这些模块和附件组成 PLC 硬件系统是我们这节要介绍的内容。表 2-2 列出了主要模块及附件的图例和功能。

表 2-2 主要模块及附件的图例和功能

| 组 件 | 功 能 | 图 例 |
|---|---|---|
| 导轨 | S7-300 的模板机架 | |
| 带总线连接器的 PROFIBUS 总线电缆 | 连接 MPI 和 PROFIBUS 子网上各个节点 | |
| 编程电缆 | 连接 CPU 和编程设备（串行接口） | |
| | 连接 CPU 和编程设备（USB 接口） | |

### 2.4.1 安装导轨（RACK）

DIN 导轨是 S7-300 可编程序控制器的机械安装铝质机架。导轨用螺钉紧固安装在墙上或机柜中，S7-300 的所有模块均直接用螺钉紧固在导轨上。即使在有可能发生机械问题的场合，有了 DIN 导轨也可以安全使用 SIMATIC S7-300 可编程序控制器。

在安装导轨时，其周围应留有足够的空间，用于散热和安装其他元器件和模块。尤其在系统中有扩展机架时，更应注意其每个机架的位置安排。

## 2.4.2 安装模块

除 CPU 模块外，每个信号模块都带有背板总线连接器，安装时先将总线连接器装在 CPU 模块的背板上，并固定在导轨上，然后依次按同样的方法将其他模块固定在导轨上。需要注意的是：一个机架上最多插 8 个 I/O 模块（信号模块、功能模块、通信处理模块等）。表 2-3 列出了 S7-300 模块的安装步骤。

表 2-3 S7-300 模块的安装步骤

| 步骤 | 连接方法 | 图例 |
|---|---|---|
| 1 | 将总线连接器插入 CPU 和信号模块/功能模块/通信模块/接口模块<br>每个模块（除 CPU 外）都有一个总线连接器<br>注意：在背板插入总线连接器时，需从 CPU 开始，取出后一个模块的总线连接器 | |
| 2 | 按照模块的规定顺序，将所有模块悬挂在导轨上（步骤 1），将模块滑到左边的模块边上（步骤 2），然后向下安装模块（步骤 3） | 2 1 3 |
| 3 | 使用 0.8 ~ 1.1nm 的螺钉旋具拧螺钉，固定所有的模块 | CPU 0.8-1.1Nm |

## 2.4.3 接线

### 1. 电源模块的连接

电源模块为系统提供了稳定的 DC 24V 工作电压源。将市电接入电源模块上的三个端子（L1、N、接地）上，再将通过两个端子（L + 和 M）输出的 DC 24V 电源线引出，为其他模块供电。具体接线如图 2-7 所示。

### 2. 信号模块的接线

前连接器用于将系统中的传感器和执行元件连接至 S7-300 PLC。模块由插入式前连接器与传感器和执行器接线，接好后插入模块（推荐使用这种顺序）。第一次插入连接器时，有

一个编码元件与之啮合，该连接器以后就只能插入同样类型的模块中。更换模块时，前连接器的接线状况无需改变就可用于同样类型的新模块。

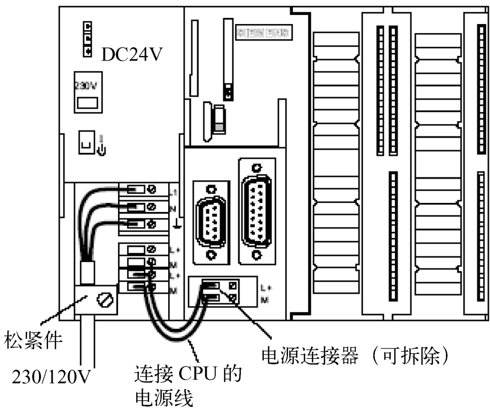



图 2-7　连接电源模块和 CPU

系统提供两种端子数量的前连接器：20 针和 40 针，分别用于具有 16 点和 32 点的模块。这两种前连接器插入模块的方法不同：20 针的前连接器上带有一个开启机构（见图 2-8）；40 针的前连接器插入只需将其上的固定螺钉拧紧即可（见图 2-9）。

在使用 SM 模块前，必须先给模块供电，否则将不能正常使用。以输入模块 SM321 DI 32×24 VDC 为例：需要把电源模块的 L+和 M 与模块的相应两个接线端子连接好。端子的分配图请参考用户手册中信号模块的部分。

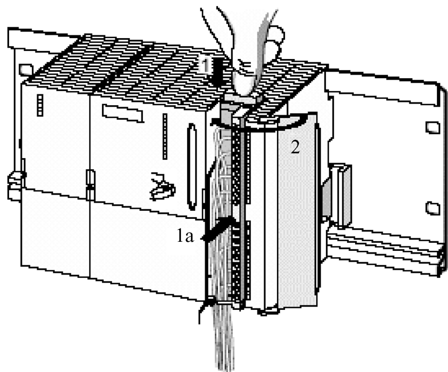



图 2-8　20 针的前连接器的插入

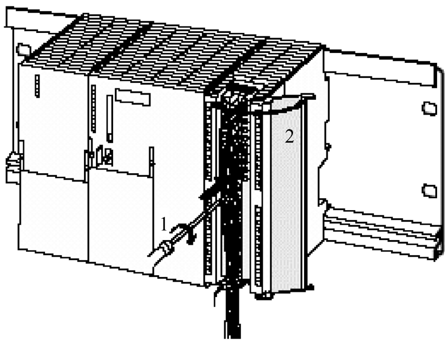



图 2-9　40 针的前连接器的插入

## 2.5　寻址

### 2.5.1　存储区中的地址及格式

如表 2-4 所示为存储器中软元件的寻址方式。当使用宽度为字或双字的绝对地址时，

应保证没有生成任何重叠的字节分配。

以 MD0 为例，双字中各字节的组合方式如图 2-10 所示。

**表 2-4　软元件的寻址方式**

| 存储区 | 名称 | 地址 | 举例 |
|---|---|---|---|
| 位存储器 | 存储位 | M | M0.0 |
| | 存储字节 | MB | MB0 |
| | 存储字 | MW | MW0 |
| | 存储双字 | MD | MD0 |
| 定时器 | | T | T21 |
| 计数器 | | C | C5 |
| 数据块 | 数据位 | DBX | DBX0.0 |
| | 数据字节 | DBB | DBB2 |
| | 数据字 | DBW | DBW4 |
| | 数据双字 | DBD | DBD4 |
| 输入过程映像区 | 输入位 | I | I2.0 |
| | 输入字节 | IB | IB2 |
| | 输入字 | IW | IW4 |
| | 输入双字 | ID | ID0 |
| 输出过程映像区 | 输出位 | Q | Q2.7 |
| | 输出字节 | QB | QB6 |
| | 输出字 | QW | QW2 |
| | 输出双字 | QD | QD4 |

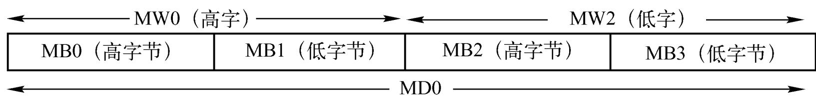



图 2-10　双字中各字节的组合方式

## 2.5.2　基于槽编址的模块地址

用户在导轨上安装好模块，STEP 7 为每个槽号指定一个确定的默认模块起始地址。图 2-11 所示为 S7-300 系统扩展最多模块时的槽号和相应的模块起始地址。数字或模拟模块具有不同的起始地址。

数字量模块的输入/输出地址由字节地址和位地址组成。例如：I0.5，其中“I”表示输入；“0”表示字节地址，取决于模块的起始地址；“5”表示位地址，是印在模块上输入点的对应数字。

模拟量模块中输入通道或输出通道的地址总是一个字地址。通道地址取决于模块的起始地址，例如 PIW256，其他通道的地址是基于起始地址并向上编址的。

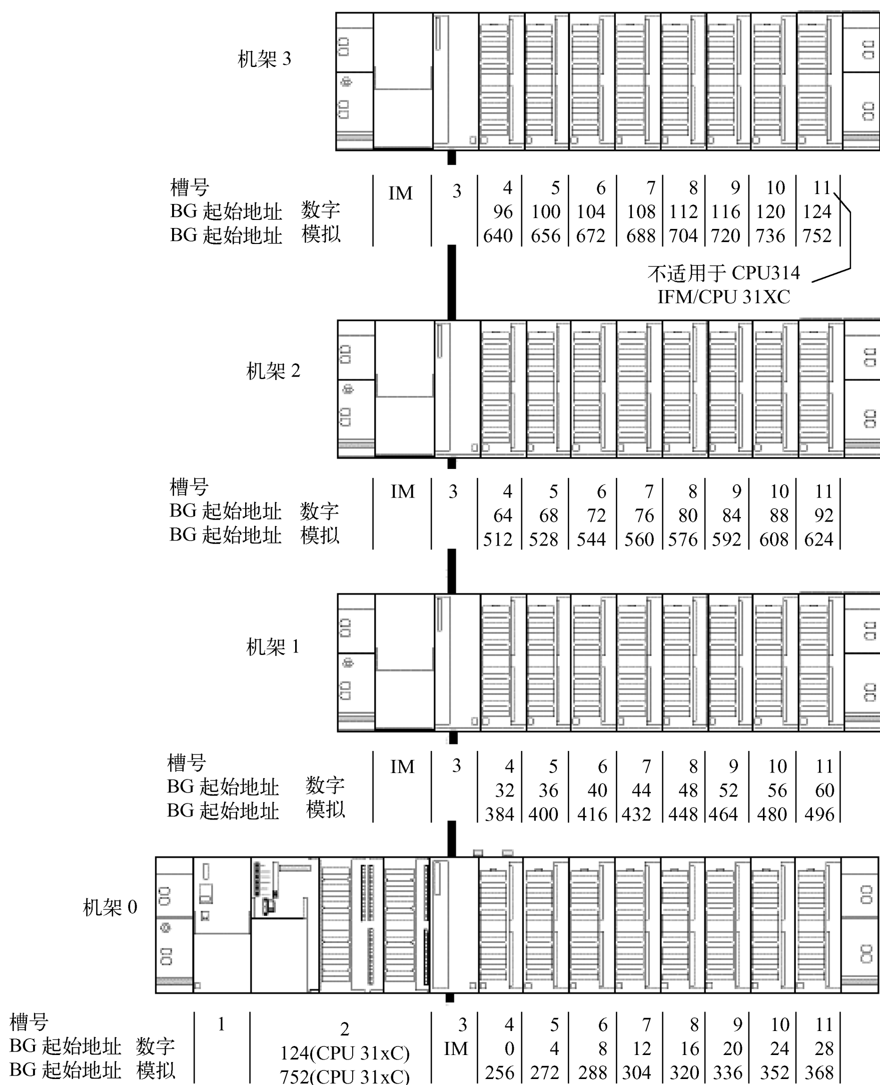



图 2-11　S7-300 系统的槽号和相应的模块起始地址

### 2.5.3　用户编址的模块地址

用户可以通过 STEP 7 软件设置所选模块的地址。这种方式的优点是用户可以直接编写独立于实际硬件组态地址的程序。具体设置方法参见第 3 章。

## 习题

1. 练习如何将 PLC 的模块安装到导轨上。

2. 一个控制系统需要 15 点数字量输入、24 点数字量输出、10 点模拟量输入和 3 点模拟量输出，试选择合适的输入/输出模块，并分配地址。

# 第 3 章　STEP 7 的使用基础

## 3.1　STEP 7 概述

STEP 7 是用于 SIMATIC 可编程序控制器的组态和编程的标准软件包，其用户接口是基于当前最新水平的人机控制工程设计，可以轻松方便地使用。STEP 7 编程软件适用于 SIMATIC S7-300/400、M7、C7 和基于 PC 的 WinAC，是供其编程、监控和参数设置的标准工具。

STEP 7 是一个强大的工程工具，用于整个项目流程的设计。从项目实施的计划配置、实施模块测试、集成测试调试到运行维护阶段，都需要不同功能的工程工具。STEP 7 工程工具包含了整个项目流程的各种功能要求：CAD/CAE 支持、硬件组态、网络组态、仿真、过程诊断等。

STEP 7 标准软件包提供一系列的应用程序（工具）：

### 1. SIMATIC 管理器

SIMATIC Manager（SIMATIC 管理器）可以集中管理一个自动化项目的所有数据，可以分布式地读/写各个项目的用户数据。其他的工具都可以在 SIMATIC 管理器中根据需要而起动。

### 2. 编程语言

用于 S7-300 和 S7-400 的编程语言梯形逻辑图（Ladder Logic）、语句表（Statement List）和功能块图（Function Block Diagram）都集成在一个标准软件包中。梯形逻辑图（LAD）是 STEP 7 编程语言的图形表达方式，它的指令语法与继电器的梯形逻辑图相似。语句表（STL）是 STEP 7 编程语言的文本表达方式，CPU 执行程序时按每一条指令一步一步地执行。功能块图（FBD）是 STEP 7 编程语言的图形表达方式，使用与布尔代数相类似的逻辑框来表达逻辑，复合功能可用逻辑框组合形式完成。

此外，还有四种编程语言作为可选软件包使用，分别是 S7 SCL（结构化控制）编程语言；S7 Graph（顺序控制）编程语言；S7 HiGraph（状态图）编程语言；S7 CFC（连续功能图）编程语言。

### 3. 硬件组态

硬件组态工具可以为自动化项目的硬件进行组态和参数设置。可以对机架上的硬件进行配置，设置其参数及属性。例如，设置 CPU 的起动特性和循环扫描时间监控。通过在对话框中提供的有效选项，系统可以防止不正确的输入。

### 4. Symbol Editor（符号编辑器）

使用 Symbol Editor（符号编辑器），可以管理所有的共享符号。其具有以下功能：可以为过程 I/O 信号、位存储和块设定符号名和注释；为符号分类；导入/导出功能可以使 STEP 7 生成的符号表供其他的 Windows 工具使用。

5. 诊断硬件

诊断硬件功能可以提供可编程序控制器的状态概况。其中可以显示符号，指示每个模板是否正常或有故障。双击故障模板，可以显示有关故障的详细信息。例如，显示关于模板的订货号、版本、名称和模板故障的状态，显示来自诊断缓存区的报文等。

6. NetPro（网络组态）

NetPro 工具用于组态通信网络连接，包括网络连接的参数设置和网络中各个通信设备的参数设置。选择系统集成的通信或功能块，可以轻松实现数据的传送。

拥有多种工具并不能保证得到很好的解决方法，关键在于这些工具之间能否实时地相互配合和协调。在运用一个工具的过程中，都使得项目数据库的内容不断增加或改变，我们希望这些改变在运用其他工具时能得到及时的更新或反映。STEP 7 就具有这种特点。比如，在符号表编辑器中添加变量名，那么打开程序编辑器窗口时，S7 程序中这些变量立即就以变量名的形式出现。也就是说，STEP 7 的数据管理能力较强，这使得其多个工具具有连续性的工作特点。在后续的章节中，将会逐渐发现 STEP 7 的这个优点。

## 3.2 安装与卸载 STEP 7

### 3.2.1 系统配置要求

1. 软件要求

STEP 7 V5.3 对操作系统有苛刻的要求，必须安装在 Microsoft Windows 2000（至少需 SP3）或 Microsoft Windows XP Professional（至少需 SP1）中，而且操作系统必须安装 Microsoft Internet Explorer 6.0 以上版本。

STEP 7 V5.4 与 Windows 操作系统的兼容性如表 3-1 所示。

表 3-1 STEP 7 V5.4 与 Windows 操作系统的兼容性

| Windows 版本 | 补丁 |
| --- | --- |
| Windows 2000 专业版 | SP4 |
| Windows XP 专业版 | SP1（SP1a）或 SP2 |
| Windows 2000 Server 2003 | 不带补丁 或 SP1 |

2. 硬件要求

STEP 7 V5.3 需要的基本硬件配置：

- PC 或编程器 PG。
- 奔腾处理器以上（CPU 主频在 600 MHz 以上）。
- RAM：256 MB，建议 512 MB。

编程器（PG）是专门为在工业环境下使用而设计的个人计算机，用于安装 SIMATIC 各设备编程时所需的软件，包括 STEP 7 等。

STEP 7 V5.4 需要的基本硬件配置：

- PC 或编程器 PG。
- 80486 处理器以上（Windows NT/2000/XP/ME 要求奔腾处理器）。

- RAM：至少 512 MB，建议 1GB。

### 3.2.2　安装 STEP 7

安装 STEP 7 是一件非常不易的事情。在安装前，请首先参考表 3-1 确认使用的操作系统与 STEP 7 V5.4 的兼容性；此外，第三方软件也可能与 STEP 7 软件产生兼容性冲突。所以，强烈建议用户在安装好操作系统后，首先安装 STEP 7。

在 STEP 7 安装光盘中，包含有 Setup 程序，使用该程序，可自动地进行安装。用户可按照屏幕弹出的指南信息的引导，一步一步地完成整个安装步骤。在光盘中附有“STEP 7 V5.4 安装及新功能介绍”PDF 文档，提供详细的安装步骤和注意事项。

在 STEP 7 安装过程中，用户可以选择三种安装范围，意义如下：

① 标准安装（推荐）：安装所有语言、所有应用程序，以及所有的举例。

② 最小安装：只安装一种语言，没有举例。

③ 用户自定义安装：用户可以选择希望安装的程序、数据库、举例和通信功能。

如果在安装过程中系统提示某些文件丢失或找不到（例如，SSF 文件找不到）或者双击 setup.exe 后安装向导无法起动，首先确保安装光盘或硬盘上的安装文件完整性；其次，如果安装文件的路径名称中含有中文字符，请将中文字符改为符号、字母或数字。

### 3.2.3　卸载 STEP 7

若用户希望安装更新版本的 STEP 7，强烈建议首先完全卸载已经安装的旧版本程序。

执行控制面板中的添加/删除程序，不能达到完全删除的目的。除此之外，还需进行注册表项删除及组件文件删除。具体操作方法可以参考西门子自动化的官方网站的链接 http://support.automation.siemens.com/CN/view/zh/23323298。

## 3.3　SIMATIC 管理器

安装好 STEP 7 后，用户可以通过两种方法打开软件：一是在桌面上双击，这个图标就是起动 STEP 7 的快捷图标，可以快速起动 STEP 7，打开 SIMATIC 管理器窗口；另一种方法是从任务栏中的“开始/SIMATIC/STEP 7”进入，在这里，用户也可以访问所安装的其他 SIMATIC 软件。

SIMATIC 管理器窗口是 STEP 7 软件的主窗口，用户可以创建和同时管理自己的多个项目。它是一个在线/离线编辑 S7 对象的图形化用户界面，这些对象包括项目、用户程序、块、硬件站和工具。利用 SIMATIC 管理器可以管理项目和库、起动 STEP 7 的多个工具、在线访问 PLC 和编辑存储器卡等。

在 SIMATIC 管理器窗口中可以同时打开已经建好的两个项目，如图 3-1 所示为项目的 SIMATIC 管理器窗口。从图中可以看出，显示出两个项目的视图，通过点击各个项目的标题栏就能轻松实现各个项目之间的切换。在项目中，数据以对象形式存储。项目中的对象按树型结构组织，这种树型结构类似于 Windows 2000 资源管理器，只是图标不同。

每个项目的视图被分成左右两个部分：左侧视图显示项目的层次结构，右侧视图显示在左侧视图中当前选中的目录下所包含的对象。在项目结构中，点击“SIMATIC 300 Station”，

右侧视图中显示这个站的硬件（Hardware）图标，双击可以打开硬件组态窗口。在项目结构中，点击“Blocks”，则这个项目中包含的软件程序块显示在右侧视图中，用户可以继续创建或删除程序块，也可以双击打开某个程序块进行编辑。

菜单栏和工具栏主要实现以下几类功能：

① 项目文件的管理。

② 编辑、插入各类对象资源。

③ 程序的下载、监控和诊断。

④ 视图、窗口排列、环境设置。

⑤ 在线帮助。




图 3-1　项目的 SIMATIC 管理器窗口

## 3.4　硬件组态

STEP 7 软件中的“HW Config（硬件组态）”为用户提供组态实际 PLC 硬件系统的编辑环境，将电源、CPU 和信号模块等设备安装到相应的机架上，并对 PLC 各个硬件模块的参数进行设置和修改。当用户需要修改模块的参数或地址，需要设置网络通信，或者需要将分布式外设连接到主站的时候，都要做硬件组态。

在 S7-200 项目中，一般直接编写程序即可，很少需要在 Micro Win 中做硬件组态，只在有需要时才进行一些硬件配置。但是，硬件组态在 S7-300/400 应用中是必要的，是完成编程的第一步。

### 3.4.1　硬件组态步骤

通过“STEP 7 Wizard:' New Project（新建项目向导）”，根据系统提示，可以便捷地完成基本的硬件组态步骤。

为了更详细地说明硬件的组态步骤，下面利用菜单“File/New”逐步建立新项目，具体步骤如下：

1）执行菜单命令“File/New”，在打开的对话框中点击“Browse”选择文件夹，选择项目的存储位置，建立一个新项目名为“zfz1”，如图 3-2 所示。

2）单击“OK”按钮后，在 SIMATIC 管理器中，只显示出一个新建立的项目名称

“zfz1”。在这个项目名上按鼠标右键选择“Insert New Object（插入新对象）”，可以看到这里供用户插入多种资源，包括 300 或 400 站、网络和程序等。这里，选择插入一个“SIMATIC 300 station”，如图 3-3 所示。

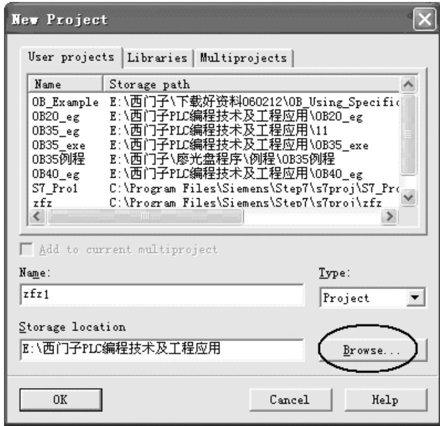



图 3-2　建立新项目




图 3-3　插入一个“SIMATIC 300 station”

3）选择插入的 300 站，双击右侧窗口中包含的 Hardware 图标，打开“HW Config（硬件组态）”窗口。这时，窗口中无任何内容，需要亲自动手逐一添加。

4）首先在右侧的硬件目录“SIMATIC 300 / RACK－300”中双击 Rail（机架），一个模拟的机架框就出现在左侧的窗口中。在这个机架上，用户可以配置具体的模块。首先在左侧视图中单击模块将要存放的位置，然后在右侧视图中双击选择的模块，或将选择的模块拖入存放的位置即可，如图 3-4 所示配置机架和具体的模块。

一般情况下，机架中的第一行放置电源模块，用户也可以让这行空着，这不影响编译，但是在实际机架中必须得有电源模块。第二行放置 CPU 模块，若 CPU 模块集成了其他功能，则会在机架中多占用一定空间。第三行放置接口模块，用于扩展更多的机架，以适应更加复杂的实际应用，所以一般为空不用。第四行之后放置其他模块，比如信号模块、通信模块等。

通过这四个步骤，我们就完成了基本的硬件组态。更重要的是，为了使每个模块按照要求工作，还要设置模块的参数。具体详见 3.4.2 节。

下面我们来观察“HW Config（硬件组态）”窗口，如图 3-5 所示，硬件组态窗口由四部分视图组成：

① 左上方视图显示了当前 PLC 站中的机架 UR，用一个可移动的表格形象地代表机架，表中的每一行代表机架中的一个插槽。

② 左下方视图显示了机架中插入模块的详细信息，包括订货号、版本、地址分配等，

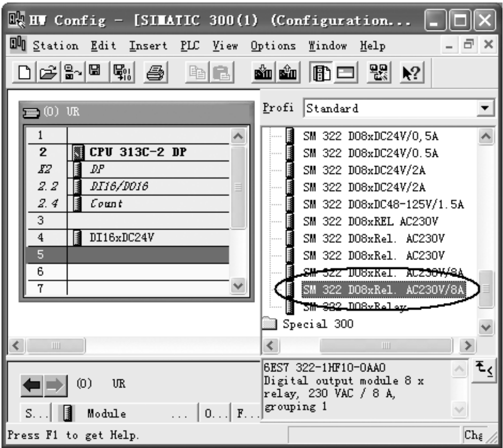



图 3-4　配置机架和具体的模块

在这里，用户可以修改网络地址和 I/O 地址等。

③ 右上方视图显示的是硬件目录，用户可以选择相应的硬件模块插入机架。在 SIMATIC 300 目录下，包含了用于组态 PLC 300 系统的所有硬件。比如，“PS－300”文件夹中包含所有的电源模块，“SM－300”文件夹中包含所有的信号模块。通过“Catalog（目录）”按钮，可以显示/隐藏硬件目录。

④ 右下方视图显示硬件目录中选中模块的信息，包括模块的功能、接口特性和对特殊功能的支持等。

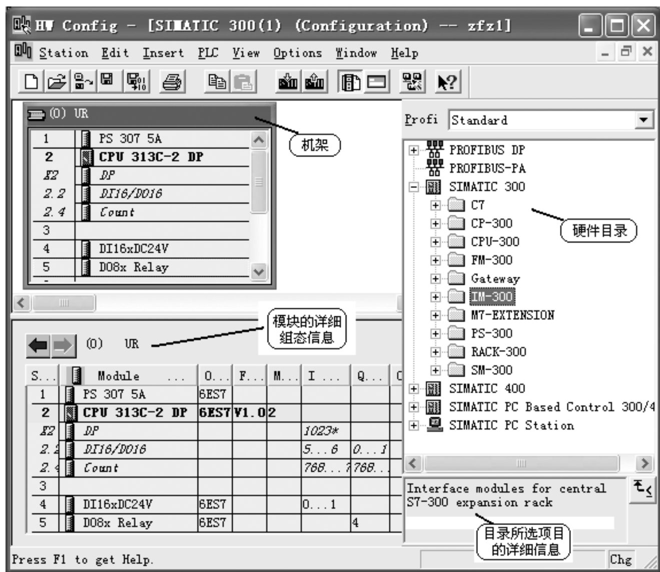



图 3-5　硬件组态窗口

## 3.4.2　参数设置

在机架和模块的详细组态信息中，双击每个模块都会弹出其“Properties（属性）”对话框，用户可以设置各类参数。

### 1. 设置 CPU 属性

双击 CPU 所在的行，弹出的对话框内容很丰富，包括多个选项卡。

“General”选项卡如图 3－6 所示，包括 CPU 的基本信息和 MPI 的接口设置。单击“Properties”按钮，用户可以选择建立 MPI 网络，并设置 MPI 通信速率等参数。

“Startup”选项卡如图 3-7 所示，可以设置 CPU 的起动特性参数。如果没有选中“Startup when expected/actual configuration differ（当预设组态与实际组态不同时起动）”，并且至少一个模块没有插入到组态时的指定槽位，或者插入的模块不是组态的模块时，CPU 将进入 STOP 状态；如果选中，即使有上述问题，CPU 也会正常起动，除了 PROFIBUS－DP 接口模块外，CPU 不会检查 I/O 组态。

对于其他选项卡，用户可根据实际需要进行设置，详细信息请参阅第 8 章。

双击 2 号槽的 DP，弹出图 3-8 所示 CPU 的“DP”选项卡，单击“Properties”按钮可以建立 PROFIBUS 网络，并设置 PROFIBUS DP 网络的参数。详见第 8 章相关内容。

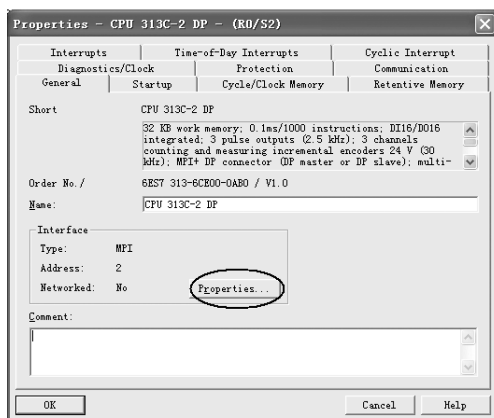



图 3-6　CPU 的“General”选项卡

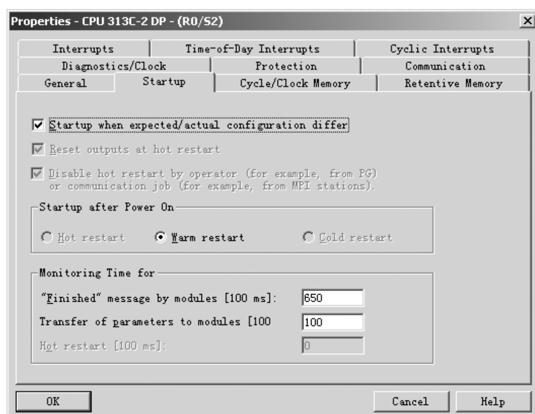



图 3-7　CPU 的“Startup”选项卡

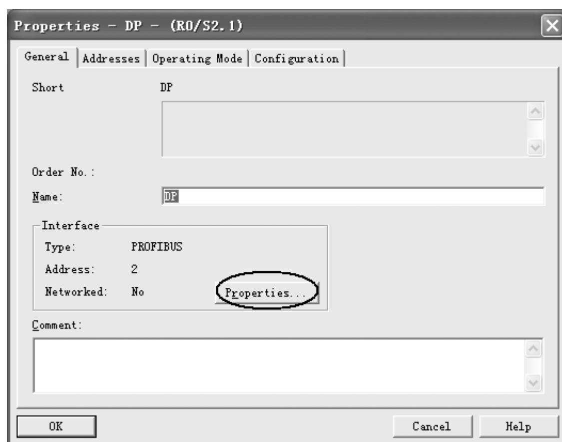



图 3-8　CPU 的“DP”选项卡

## 2. 设置 SM 的属性

除了对 CPU 的设置，SM（信号模块）的设置是更为常用的。SM 模块包括数字量模块（DI/DO）和模拟量模块（AI/AO），还有一些紧凑型的 CPU 也集成了 DI/DO 模块，比如 313C-2DP 集成了 16 点的 DI 和 16 点的 DO，如图 3-5 中的“2.2 行”。

对于 SM 的参数设置方法与设置 CPU 参数一致，双击模块所在的行即可。在图 3-5 中双击 2.2 行中的“DI16/DO16”，打开属性对话框，如图 3-9 所示，在“Addresses”选项卡中，取消“System Selection”选项后，就可以在“Start”位置修改模块的默认地址，并键入新地址。而且，这些地址不仅在内存中有映像，而且与实际的物理接口相对应，在编程中也要对应用这些地址。注意，各个模块的地址是不能重合的。

对于模拟量模块的属性设置，以 SM331（AI8×12bit）模块为例，模拟量转换与 PLC 的接口地址是从字节 256 开始，到 271 结束，共 16 个字节，每个输入通道占用 2 个字节，如通道 0 转换的结果存入到 PIW256 中，通道 1 的转换结果存在 PIW258 内，依次类推，通道 8 的转换结果存入 PIW270 内。单击属性设置的 Inputs 选项卡，如图 3-10 所示，根据传感器输出的信号类型，可以在 Measuring 域内设置通道类型，如通道 0、1 为 2DMU（D）；设置通道 2、3 为 4DMU（C）；设置通道 4、5 为 1～5V（B）；设置通道 6、7 为 +/－80 mV

(A)。在 Enable 域内选择两个复选框，允许诊断和硬件中断。设置完成后单击“OK”按钮。

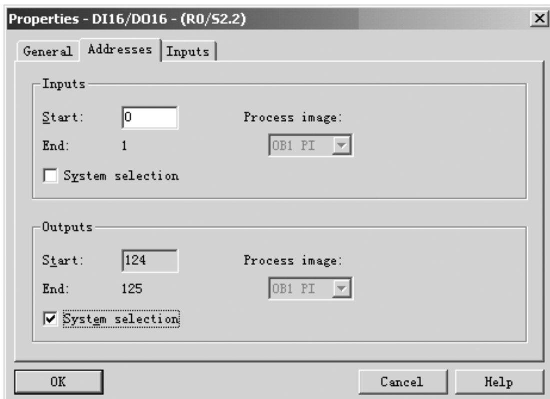



图 3-9　修改模块的默认地址

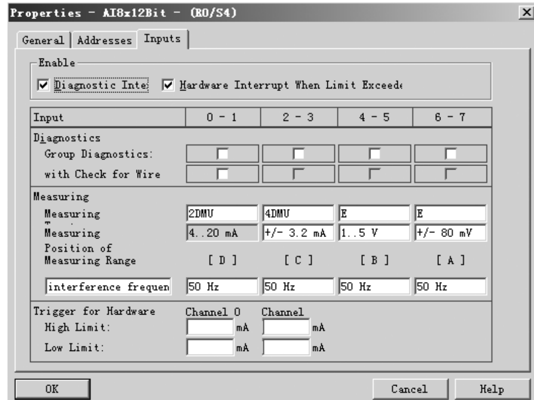



图 3-10　模拟量模块输入通道属性设置

### 3.4.3　硬件组态目录的更新

SIMATIC 是一个庞大的家族。它的每一个软硬件成员都在不断地发展。在硬件组态时，机架上模块的订货号必须与实际物理模块的订货号一致，而且一个型号的 PLC 模块可能有多个订货号。例如：CPU313C-2DP 目前为止有两种不同的订货号，分别是“6ES7-6CE00-0AB0”和“6ES7-6CE01-0AB0”，在 STEP 7 V5.2 中，只支持第一种，而在 STEP 7 V5.3 中，两种订货号均支持。如果用户不想付费重新购买 STEP 7 V5.3，那么上网更新 STEP 7 V5.2 的硬件目录就是最好的选择。

在 Internet 上更新硬件目录的步骤如下：

1）在硬件组态窗口中，选择菜单“Options/Install HW updates”，就开启硬件更新了。当用户第一次使用时会提示用户设置下载网址（已存有默认网址）和更新文件的保存目录。

2）设置完毕后，弹出如图 3-11 所示的安装硬件更新对话框，选择“Download from Internet”，单击“Execute”按钮，就可以下载最新的硬件列表。

3）弹出“Download hardware updates”对话框，选择需要的硬件，单击“Download”按钮即可。

4）下载完成后，提示用户安装更新。选中需要更新的硬件，而且“Installed”一栏显示为“no”，表示没有安装，那么单击“Install”，按照提示即可完成更新。

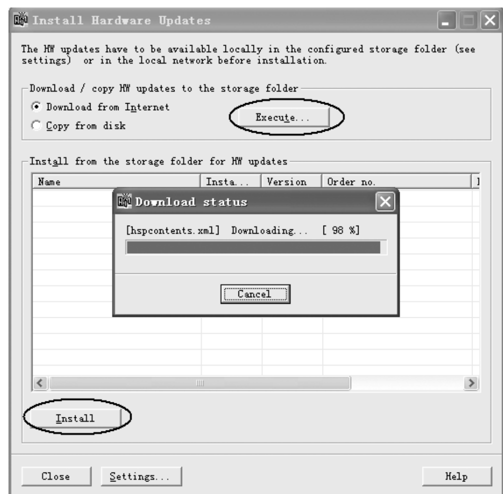



图 3-11　安装硬件更新对话框

也可以上网下载新模块的 GSD 文件，再加载到 STEP 7 软件中。方法如下：GSD 文件下载后解压，打开 Step 7，进入硬件组态窗口，

选择菜单“Options / Install GSD”，然后单击“Browse”按钮找到刚才解压的目录，会显示若干文件，有 GSE 文件首选 GSE，没有就选择 GSD，只要选一个即可，然后单击“Install”按钮完成更新。

## 3.5 软件编程

### 3.5.1 程序编辑器界面

在 SIMATIC 管理器窗口中，选择项目结构中的“Blocks”，则在右视图中单击鼠标右键打开快捷菜单，执行菜单命令“Insert New Object”，可以插入以下资源：

1）Organization Block（组织块）。

2）Function Block（功能块）。

3）Function（功能）。

4）Data Block（数据块）。

5）Data Type（用户定义的数据类型）。

6）Variable Table（变量表）。

双击“Blocks”下的任何一个程序块，都会打开程序编辑器界面，如图 3-12 所示为 FC6 的程序编辑器的语句表界面，由四个视图组成，分别为编程元件窗口、局部变量声明窗口、代码窗口和信息窗口。

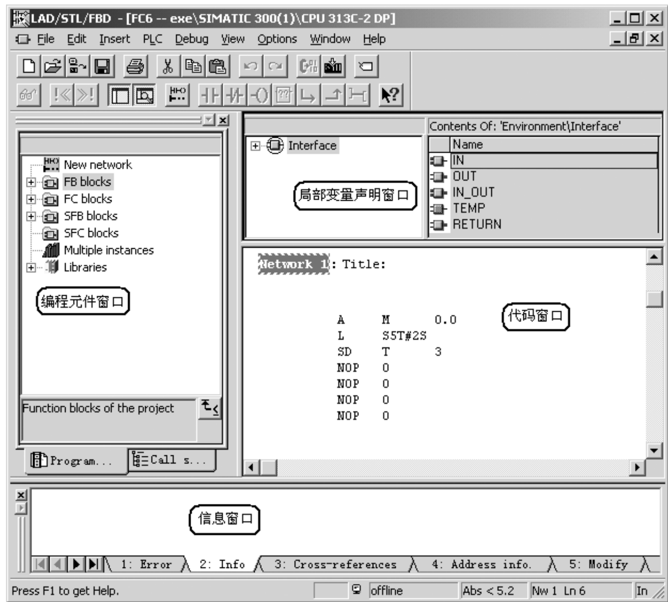



图 3-12 程序编辑器的语句表界面

### 3.5.2 使用程序编辑器

如前所述，STEP 7 支持三种基本的语言：STL（语句表）、LAD（梯形图）和 FBD（功

能块图）。梯形图是使用的最多的 PLC 图形编程语言，所以以 LAD 编程为例，介绍程序编辑器的使用方法。可以通过菜单“View/LAD”选择当前的编程环境为 LAD。

在代码窗口中，插入编程元件的方法是：先单击网络（Network n）中的长横线，它将变为深绿色粗线，此时在编程元件窗口中，双击将要插入的编程元件或拖拽编程元件到网络中即可。通过单击工具栏中的按钮和，可以建立串并联分支。

执行菜单命令“Option/Reference Data”，可以打开参考数据（用法详见第 5 章）。执行菜单命令“Option/Symbols Editor”，可以打开符号表。执行菜单命令“Option/Customize”，可以根据用户的喜好设计代码窗口的风格，比如字体的大小及颜色等。

试试分别执行菜单命令“View/Display with”中的五个选项，观察代码窗口中的元素有什么改变？

### 3.5.3 变量与符号

在 STEP 7 中，用户可以直接使用的变量包括 PLC 的输入/输出（I/O）地址、M 存储区地址、数据 DB 块名、功能块名和系统已经在组织块和逻辑块中定义的变量等，如：Q0.0、M20、FC2、OB40。如果能在 STEP 7 中将变量用具有实际意义的符号名字代替，那么用户程序的可读性就会更好，例如，将项目中控制电源总开关的变量赋给地址 I0.0，那么可以为 I0.0 定义符号 POWER_ ON。

**1. 全局符号和局部符号**

STEP 7 中可以定义两类符号：全局符号和局部符号。与其他编程语言的定义一致，全局符号在整个用户程序范围内有效；局部符号仅仅在定义的块内部有效。如表 3-2 所示为全局符号和局部符号的区别。

**表 3-2　全局符号和局部符号的区别**

| | 全局符号 | 局部符号 |
|---|---|---|
| 有效范围 | 在整个用户程序中有效，可以被所有的块使用，在所有的块中含义是一样的 | 只在定义的块中有效，相同的符号可在不同的块中用于不同的目的 |
| 允许使用的字符 | 字母、数字及特殊字符（除 0x00、0xFF 及引号以外的强调号），如使用特殊字符，则符号须写在引号内 | 字母<br>数字<br>下划线（_） |
| 使用对象 | 可以为以下各项定义：<br>I/O 信号（I，IB，IW，ID，Q，QB，QW，QD）<br>I/Q 输入与输出（PI，PQ）<br>存储位（M，MB，MW，MD）<br>定时器（T）/计数器（C）<br>逻辑块（FB，FC，SFB，SFC）<br>数据块（DB）<br>用户定义数据类型（UDT）<br>变量表（VAT） | 可以为以下各项定义：<br>块参数（输入，输出及输入/输出参数）<br>块的静态数据<br>块的临时数据 |
| 定义范围 | 符号表 | 块的变量声明表 |

**2. 符号的定义**

符号名的长度不能多于 24 个字符，而且定义时不区别大小写字符。需要注意的是：数据块中的地址（DBD、DBW、DBB、DBX）不能在符号表中定义，应在数据块声明表中定义。组织块（OB）、某些系统功能块（SFB）和系统功能（SFC）已被系统根据块的功能预

先赋予了符号名。

在符号表中可以定义全局符号，项目中符号表的位置如图 3-13 所示，在 SMATIC 管理器窗口的左视图中，单击“S7 Program（1）”，在右视图中就自动出现了符号表“Symbols”，双击打开符号表的编辑界面，如图 3-14 所示。在符号表的空白行中输入符号名和地址，可定义一个新符号。在第二列“Status（状态）”中，若出现红色的叉号，表明在此行的变量没有被定义好。如果用户希望删除一个已经定义的符号，可以将鼠标移至该行的标号处，单击即选中符号所在的这一行，然后按〈Delete〉键。

需要注意的是，并不需要下载符号表。编辑符号后并保存符号表，这时符号表就生效了。查看菜单命令“View / Display”，确认激活“Symbolic Representation（符号表达方式）”，用户就可以在程序中看到地址已经被其符号名所代替了。

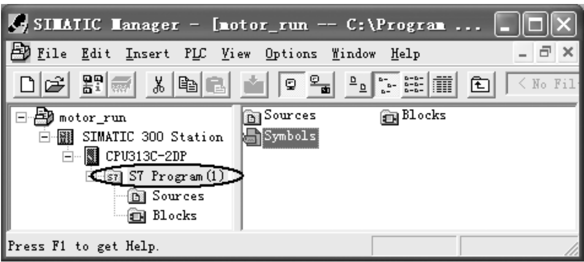
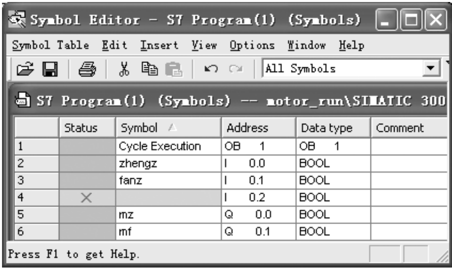



图 3-13　项目中符号表的位置




| | Status | Symbol | Address | Data type | Comment |
|---|---|---|---|---|---|
| 1 | | Cycle Execution | OB 1 | OB 1 | |
| 2 | | zhengz | I 0.0 | BOOL | |
| 3 | | fanz | I 0.1 | BOOL | |
| 4 | × | | I 0.2 | BOOL | |
| 5 | | mz | Q 0.0 | BOOL | |
| 6 | | mf | Q 0.1 | BOOL | |

图 3-14　符号表的编辑界面

在符号表编辑器中，通过菜单“Symbol Editor/import... export...”用户可以导出当前的符号表到一个文本文件，这样就可以用文本编辑器对符号进行编辑，还可以将文本编辑器中的符号表导入到用户程序中。这种导入功能大大节省了用户的工作。导出符号表时，用户如选择文件格式为“*.DIF”，则可以在 Microsoft Excel 中打开、编辑并存储 DIF 文件；如选择文件格式为“*.SDF”，则可以在 Microsoft Access 中打开、编辑并存储 SDF 文件。

在程序块的变量声明窗口中可以定义局部符号。在功能 FC 中，可定义五种类型的局部符号，如图 3-15 所示，分别如下：

① IN（输入变量）。由调用它的块提供的输入参数。

② OUT（输出变量）。返回给调用它的块的输出参数。

③ IN_ OUT（输入_输出变量）。初值由调用它的块提供，被子程序修改后返回给调用它的块。

④ TEMP（临时变量）。暂时保存在局部数据区中的变量。在 OB1 中，局部变量表只包含这类变量。

⑤ RETURN（返回变量）。调用后的返回值。

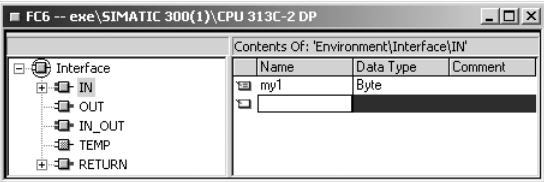



图 3-15　局部符号的定义

## 3.6 硬件接口和下载

### 3.6.1 硬件接口

**1. MPI 接口**

几乎所有的 S7-300/400 PLC 都集成有 MPI 接口，它们既是通信接口，同时也是编程接口。因此通过 MPI 下载项目是一种比较普遍且简便的方法。

PC/MPI 适配器使用户能在编程设备（即 PC）和 PLC 之间建立数据联系，通常称它为编程电缆。一般根据与编程设备的接口不同有两种适配器（见第 2 章表 2-2），一种是 RS-232 接口，另一种是 USB 接口。适配器一端接 PLC 的 MPI 接口，另一端接编程设备的 RS-232 接口或 USB 接口。

编程电缆的价格较高，虽然如此，MPI 下载方式还是在许多不易下载的场合尤其是首次下载中发挥了重要作用。

**2. PROFIBUS 接口**

若在 PC 上安装通信卡，就能使编程设备和 PLC 之间通过网络进行通信。SIEMENS 提供了 CP 5611 卡（PCI 卡）、CP5511 或 CP 5512 卡（PCMCIA 卡），将网卡安装在编程设备中，可以使 PC 连接到 MPI 或 PROFIBUS 网络中。

此时，编程电缆使用紫色的 PROFIBUS 电缆线，并需配相应的 PROFIBUS 接头。

**3. 以太网接口**

工业以太网通信卡，如 CP 1512 卡（PCMCIA 卡）或 CP 1612 卡（PCI 卡），可以将 PC 连接到以太网中。在要求不是很高的场合，也可以用普通网卡代替工业以太网通信卡。

以太网是一种非常方便而且高速的下载方式。使用以太网路径下载的前提是网络内或者下载目的站集成有以太网接口，或者通过总线扩展了 CP343 - 1/CP443 - 1 等以太网模块。一般的 PLC 自身是没有以太网接口的，但是带“PN”的 PLC 本身集成了以太网接口，使用以太网下载硬件组态或程序非常方便，比如 315 PN/DP。另外，使用以太网不仅可以下载项目，还可以在网络通信时，对网络内的各个节点同时进行监控。这是使用 MPI 下载方式无法实现的。

### 3.6.2 下载方法

用户如何把编程器或 PC 的硬件设置和用户程序传到 PLC 呢？这就是所谓的下载。

由上可知，根据实际需求，用户可以选择不同的接口下载项目。那么，如何能顺利使用好这些接口完成下载工作呢？基本的思想是软件设置必须与硬件接口配合。这里的软件设置指在“Setting PG/PC Interface（设置 PG/PC 接口）”窗口中的设置，通过在控制面板中双击“Setting PG/PC Interface”图标打开；或者可以在 SIMATIC 管理器窗口中，执行菜单命令“Option/ Setting PG/PC Interface...”打开。只有在安装好 STEP 7 软件的 PC 中其“控制面板”内才会出现这个设置项目。

**1. 下载原则**

在下载整个项目之前，最好先下载硬件组态。因为下载硬件组态的时候，STEP 7 都会

提示用户指定即将下载的目的站点，而下载整个项目或是单独下载程序的时候会默认硬件组态时指定的目的站地址进行下载。

**2. 下载条件**

（1）CPU 必须在允许下载的工作模式下（STOP 或 RUN）

建议用户在“STOP”工作模式下载。在“RUN”工作模式下，程序将一次下载一个块。如果重写一个旧的 CPU 程序，可能会出现冲突。当处理循环时，CPU 就会进入 STOP 模式。

（2）编程设备和 CPU 之间必须有一个连接

最常用的连接就是编程电缆了，此外还包括 PROFIBUS – DP 电缆和工业以太网的网线等。要使用户能有效访问到 PLC，不仅需要实际的物理连接，还需要设置好“控制面板”中的“Setting PG/PC Interface（设置 PG/PC 接口）”。

（3）用户已经编译好将要下载的程序和硬件组态

建议用户养成良好的习惯，最好能在编译完后及时保存，再下载到 PLC。这样就可以保证编程设备中的程序和 PLC 中程序的一致性。“保存”的作用是将编程设备中当前的软硬件内容保存在编程设备的硬盘上；而“下载”的作用是将编程设备中当前的软硬件内容下载到 PLC 中。这是两个不同的概念。尤其是当用户在线调试时，用户会在线修改程序内容，这时一定要先保存程序再下载，避免下载的程序与最终保存的程序版本不一致。

**3. 下载的基本方法**

在 SMATIC 管理器窗口、硬件组态窗口和“LAD/STL/FBD”窗口的工具栏上，都有下载工具按钮，而且在这些窗口的菜单项中也含有下载选项“PLC/Download”。在下载新的全部用户程序之前，应该执行一次 CPU 存储器的复位。

1）在 SMATIC 管理器窗口中，首先在左视图或右视图中选中要下载的对象，包括项目、PLC 站、程序块等，然后单击下载工具按钮即可。Microsoft 公司软件中的〈Shift〉和〈Ctrl〉键的用法在 STEP 7 中同样适用，便于用户同时选择多个对象。

2）在硬件组态窗口中，用户组态好硬件后，先单击按钮编译并保存当前的硬件组态，然后单击按钮下载即可。

3）在“LAD/STL/FBD”窗口中，单击按钮下载的是当前窗口中编译好的程序。

注意：如果 S7 程序是硬件站的一部分，用户可以在块的文件夹中发现一个“System data”（系统数据）符号。它包含组态数据和参数分配数据，下载时确认“Do you want to load the system data?”信息，选择“Yes”按钮，即要下载这些数据。如果 CPU 处于“RUN”方式，会弹出一个信息窗口，要求自动把 CPU 切换到停机状态。在完成下载后，CPU 又自动会转为 RUN 模式。这样就不必频繁操作开关了。

**4. MPI 下载**

（1）物理连接

用黑色 MPI 下载电缆连接 PC 的 USB 口与 PLC 的 MPI 接口。

（2）设置 PG/PC 接口

在控制面板→“设置 PG/PC 接口”→“应用程序访问点”内，选择“S7ONLINE（STEP7）”，在“已使用的接口参数分配”下选择下载路径“PC Adapter（MPI）”，如图 3–16 所示。点击“属性”按钮，设置通信速率为 187.5Kbit/s，MPI 网的最高站点地址设为 31，

如图 3-17 所示；在“本地连接”选项卡中选择 PC 的接口为 USB（若使用的下载电缆与 PC 连接端为 COM 口，则选择 COM）。点击“确定”按钮完成操作。

（3）下载硬件组态

打开项目的硬件组态，点击下载按钮，在随后出现的几个对话框内选择“确定”按钮后，出现如图 3-18 所示的“选择节点地址”对话框。其中：

① 1 号框用来指定即将要下载硬件组态的目的站（MPI）。

② 2 号框用来搜索所有通过“PC Adapter（MPI）”电缆与 PC 连接的可访问的节点。

首先点击“视图”，等待出现目标站点后，用鼠标选择该站点，然后点击“确定”按钮，即可将组态下载到目的 PLC 内。

如果还有其他的组态要下载到网络中另外的节点内，则将黑色 MPI 下载电缆转接到另一块 PLC 的 MPI 接口上。重复上述步骤，可将该站的组态下载到另一个 PLC 内。




图 3-16　选择访问路径




图 3-17　设定通信参数




图 3-18　指定下载的目标站

（4）下载程序

下载完硬件组态，用户可以在“SIMATIC Manager”内选择要下载程序所在站的“Blocks”，然后直接点击“下载”按钮就可以将该块内所有的程序全部下载到默认的节点内，该节点是通过下载硬件组态时指定的。

下载块后，如果用户后来仅仅修改了某个程序，则可以单独下载该程序。

当然也可以在“SIMATIC Manager”下选择站点名称，然后直接点击“下载”按钮，则系统就以硬件组态时指定的节点为默认节点，将整个站点（包括硬件组态和程序）下载下去。

需要指出的是，如果用户的 PG/PC 装有 CP5611、CP5511 等 PCI 智能卡，则访问路径除了选择“PC Adapter（MPI）”外，还可以选择“CP5611（MPI）”等路径来下载项目。“CP5611（MPI）”通信速率等参数的配置与“PC Adapter（MPI）”类似。“CP5611（MPI）”路径下具有网络诊断功能，利用它可以诊断该路径下物理线路的连接情况。诊断结果可参考图 3-19。图中框内打勾的为激活的站点，没有打勾的但是高亮显示的为从属站点，其中“1”为 PC 的地址，“2”和“3”分别是 PLC 的地址，“2”组态为主站，“3”组态为从站。不论使用哪种协议下载，本地地址（即编程器或者 PC）都是默认的 0，用户不要改变它。

5. PROFIBUS 下载

如果用户的 PG/PC 装有 CP5611、CP5511 等 PCI 智能网卡，则访问路径可以选择“CP5611（PROFIBUS）”，如图 3-20 所示。在下载之前可以先诊断网络线路的连接情况，诊断情况也可以参考图 3-19。项目下载的步骤与 MPI 下载相关内容描述类似，用户只需按照系统提示依次进行即可。

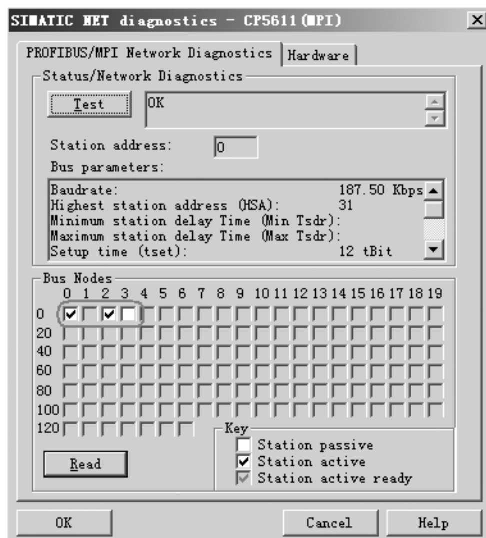



图 3-19　CP5611（MPI）的诊断功能

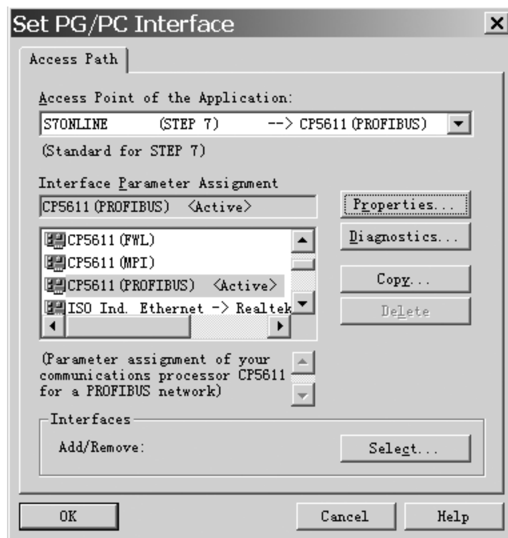



图 3-20　PROFIBUS 访问路径

6. 通过 TCP/IP 下载

（1）线路物理连接

如果既要下载项目又想在网络通信时对各个节点同时进行监控，则最好准备一个交换机，并用以太网线将网络内的各个节点连接到交换机的 RJ45 接口。如果节点处没有集成

RJ45 接口，则需要扩展以太网模块，如 CP343 - 1、CP443 - 1 等。

如果只是下载项目，不去监控网络中各个节点的通信情况，则可以使用交叉线直接连接 PC 和 PLC。

（2）设定 IP 地址

设定 IP 地址的目的是让 PC 和各个 PLC 站处于同一个子网内，STEP 7 内在硬件组态界面下 PN 接口或者以太网模块的 IP 设定可以参考图 3-21 和图 3-22。

双击图 3-21 中的“PN - IO”，在随后出现的对话框内点击“属性”按钮，出现如图 3-23 所示的对话框。在框内填入 IP 地址及默认的子网掩码，如“192. 168. 10. 1 ”、“255. 255. 255. 0”。这里不需要新建以太网。

对其他站点的以太网接口的 IP 地址的设定也参照上述方法执行。需要指出的是之所以使用“192. 168. . . ”，是因为该段内 IP 地址可以不被外网所路由。用户对局域网内所有的节点设定的 IP 地址必须是在同一个子网内，并且不能重复。

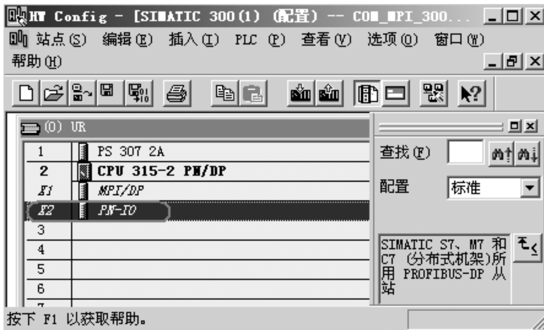



图 3-21　配置以太网接口 PN - IO




图 3-22　指定节点的 IP 地址

PC 的 IP 地址设定方法如图 3-23 所示。原则上只有当 PC 与其他的网络节点处于同一个子网内时，才能建立通信。也就是说在下载硬件组态之前，用户最好应该设法知道目的站点的 IP 地址，否则通常情况下首次用 TCP/IP 不容易将组态下载到目的 PLC 内。而一旦成功下载了硬件组态（比如使用 MPI 下载），PC 的 IP 地址就应该改为与硬件组态内设定的 IP 在同一个子网内。这时候就可以如图 3-23 所示那样设置 PC 的 IP 地址为“192. 168. 10. 100”，子网掩码使用默认的“255. 255. 255. 0”，默认网关为“192. 168. 10. 254”。

（3）设置 PG/PC 接口

在“设置 PG/PC 接口”界面修改访问路径为“S7ONLINE（STEP7）→TCP/IP→Realtek RTL8139 Family. . . . ”，其中“Realtek RTL8139 Family”为用户 PC 安装的网卡驱动名称，如图 3-24 所示。

（4）下载硬件组态与程序

打开“SIMATIC Manager”内某个站的硬件组态，点击下载按钮，然后参考 MPI 下载步骤进行硬件组态的下载。所不同的是在图 3-18 处，在点击“视图”后，通过“TCP/IP”协议找到的站地址是 IP 地址和 MAC 地址。同理，选择一个目的站点将相应的组态下载进去。

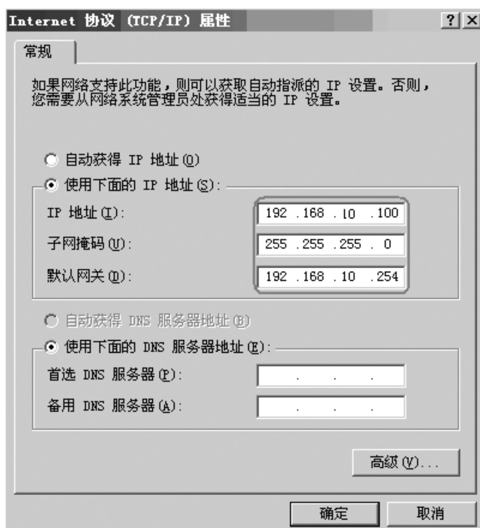



图 3-23　PC 的 IP 地址设定

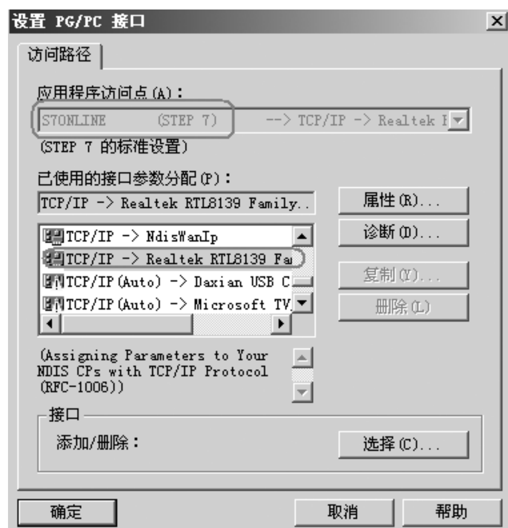



图 3-24　选择“TCP/IP”访问路径

然后在“SIMATIC Manager”下选中站点名称，点击“下载”按钮，将整个站点的组态和程序一起下载到目的节点内。这样就可以采用以太网的方式进行后续的下载和监控了。

### 3.6.3　上传

数据的反方向传输就是上传。上传的目的是将来自现场 PLC 中的信息上传到 PC 并保存在硬盘中，这些信息包括硬件组态和所有程序。

主要包括以下 3 种上传方法：

① 在 SIMATIC 管理器窗口中，工具栏内没有上传按钮，只能通过菜单“PLC/Upload Station to PG”，将一个 PLC 站的内容上传到编程设备中。编程设备是直接或者通过网络连接到 PLC 上的，上传的内容包括这个 PLC 站的硬件组态和用户程序。

② 在硬件组态窗口中，通过工具栏中的上传按钮或者菜单“Upload”上传数据。这种方式会在项目中插入一个站，但是只包括硬件组态，不包括 PLC 中的用户程序。

③ 在线状态下，可以有选择的上传用户程序。在 SMATIC 管理器窗口中，单击在线按钮打开在线窗口，选中要上传的程序块，通过菜单“PLC/Upload to PG”把选中的程序块上传到编程设备中。

## 3.7　程序归档

为了便于用户保存一个完整的项目，可以利用项目归档功能。将项目或库以压缩的形式（ARJ 或 ZIP）存入一个归档文件中，这样一个压缩的项目文件便不能被随意编辑修改。如果希望对它们进行编辑，就必须将数据解压缩，即恢复项目或库。

使用菜单命令“ File/Archive... ”对项目或库进行归档；使用菜单命令“ File/ Retrieve... ”对项目或库进行恢复。

## 3.8 如何使用 STEP 7 软件的在线帮助

STEP 7 软件具有强大的帮助系统，就好像是一部 STEP 7 软件的百科全书。不过 STEP 7 的在线帮助中全部是英文解释，用户需要有较好的英语水平。这样，用户就可以随时查阅这本百科全书，得到有效的帮助。

### 3.8.1 查找某个关键字或功能

在 STEP 7 的主界面 SIMATIC 管理器中，点击下拉菜单“Help/Contents”打开 STEP 7 的在线帮助。可以利用 Index 进行关键字的查找，或者利用 Search 进行相关搜索。

### 3.8.2 了解某个逻辑块 FB/FC/SFB/SFC 的功能及管脚的定义

在 LAD 程序编辑器中，首先单击工具栏中按钮并键入具体的逻辑块名将一个逻辑块调入到 Network 中，然后选中该逻辑块（用鼠标点击该逻辑块，外框变为绿色），按计算机键盘上的〈F1〉功能键，这时就自动显示出关于该逻辑块的功能及管脚定义的描述。用户可以在该帮助信息中了解到该逻辑块的功能、参数的描述及所要求的数据类型、可能的错误信息等，有些帮助信息中还有实用的例子程序。

### 3.8.3 应用方法

在 STEP 7 软件的在线帮助中提供了某些 SFC/FC 的调用例程。

查找例程的方法如下：将逻辑块添加到某一个 Network 中，按下计算机键盘上的〈F1〉功能键，在帮助信息中，一般会有例子程序，以及该程序的描述。

下面以使用 SFC51 为例，介绍如何使用 STEP 7 帮助中的例程。

打开空的 OB1 程序，然后在编程元件窗口选“Libraries /Standard Libraries/System Function Blocks /SFC51 RDSYSST DIAGNSTC”，插入到 Network 中。按〈F1〉键，出现 SFC51 的在线帮助信息。可具体读一下信息的内容，在信息的最底部点击“Example for module diagnostics with the SFC51”，然后选择点击“STL Source File”，选中全部 STL Source 源程序复制到 STEP 7 程序的 STL Source（1）中，存盘并编译此源程序，提示没有错误。最后，重新回到 Blocks 下，观察到此时已经生成 OB1、OB82、DB13 和 SFC51 程序块了。也就是说，帮助系统中是通过“STL Source File”直接给出例程的。这时，用户就可以参考这个例程了。

## 习题

1. 若实际导轨上的 CPU 模块为 CPU 315-2DP（订货号是 6ES7-2AF00 – 0AB0），DI/DO 模块为（订货号为 6ES7 323 – 1BL00 – 0AB0），试在 STEP 7 中进行硬件组态。

2. 将 PLC 中的信息保存到 PC 的新建项目“Mynew”中。

# 第 4 章　编 程 语 言

## 4.1　概述

IEC（国际电工委员会）是为电子技术领域制定全球标准的世界性组织，IEC61131 则是 PLC 的国际标准。它由 5 部分组成：通用信息、设备要求与测试、编程语言、用户指南、通信服务规范。其中 IEC61131 -3 是 PLC 的编程语言标准，著名的自动化设备制造商如西门子、罗克韦尔、施奈德等公司均推出与其兼容的产品。

STEP 7 是西门子公司 S7 -300/400 系列 PLC 的编程软件。在 STEP 7 软件包中配备了三种基本编程语言：梯形图（LAD）、语句表（STL）和功能块图（FBD）。当然，三种语言各有其特点。对于初学者，梯形图和功能图相对而言更易上手。而语句表则功能强大，更符合编程思路，可以实现某些复杂的功能，如间接寻址、压栈、出栈等，这些是梯形图和功能图无法实现的。三种编程语言在 STEP 7 中有近 90% 以上的语句可以互相转换。

梯形图：与继电器电路图相似，具有直观易懂的特点，适合数字量逻辑控制。梯形图程序举例如图 4-1 所示。

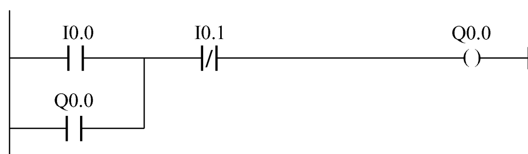



图 4-1　梯形图程序举例

语句表：类似微机中的汇编语言，是由多条语句组成一个程序段，功能强大。在运行时间和要求的存储空间上功能最优，尤其在数学运算和通信设计方面更为适合。上述梯形图的功能在语句表中实现，形式如下：

```
A(
O     I     0.0
O     Q     0.0
)
AN    I     0.1
=     Q     0.0
```

功能块图：类似布尔代数的图形逻辑符号，熟悉数字电路的人非常适合。实现如图 4-1 功能的功能块图程序举例如图 4-2 所示。

梯形图、语句表、功能块图具备完备的指令系统，支持结构化编程方法。任意三种语言用户可以随意选择，主要取决于工作需要和个人擅长。

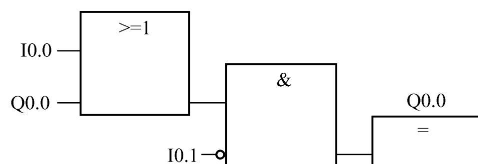



图 4-2　功能块图程序举例

STEP 7 中除了三种基本的编程语言可供使用外，还有一些更高级的编程语言，如顺序功能图（SFC）、结构文本（ST）、S7 HiGraph 编程语言、S7 CFC 编程语言。其中顺序功能图用于编制顺序控制程序。在这种语言中，工艺流程可划分为若干个顺序步骤，可以非常清晰地表述其顺序控制过程。而另一种编程语言 S7 HiGraph 的编程思路同样可以借鉴。这种语言使用状态图（State graph）来描述异步、非顺序过程。这与 Verilog 硬件描述语言中的状态机的概念类似。状态图将系统划分为几个单元，每个单元有不同的状态，而不同的状态在满足一定转换条件下可以互相转换。在以后的章节中我们将主要介绍这两种编程思想。

STEP 7 中的指令系统主要包括位操作、定时器/计数器、数学运算、数据处理等内容。下面将从程序结构入手，先搭框架，然后再填充细节。

## 4.2　STEP 7 编程语言的程序结构

为了便于阅读和理解，在编程中常常将程序分成若干部分。STEP 7 编程语言支持这种划分，并且提供了必要的功能。每个程序部分具有其技术和功能基础，我们称之为“块”。

块是程序中真正有用的部分，包括用户块和系统块，它们在功能、使用方法和结构上各不相同。

### 4.2.1　用户块

根据逻辑功能的不同，用户块分为组织块（OB）、功能块（FB）、功能（FC）和数据块（DB）。

**1. 组织块（OB）**

组织块是操作系统和用户程序之间的接口。组织块只能由操作系统来起动。各种组织块由不同的事件起动，且具有不同的优先级，而循环执行的主程序则在组织块 OB1 中。

**2. 功能块（FB）**

功能块是通过数据块参数来调用的。它们有一个放在数据块中的变量存储区，而数据块是与其功能块相关联的，称为背景数据块。当然，每一个功能块可以有不同的数据块。这些数据块虽然具有相同的数据结构，但具体数值可以不同。就像 C 语言中的函数，其形参可以相同，但带入函数的实参却可以不同。

3. 功能（FC）

功能没有指定的数据块，因而不能存储信息。功能常常用于编制重复发生且复杂的自动化过程。

4. 数据块（DB）

数据块中包含程序所使用的数据。在编制数据块时，用户可以决定数据的类型、格式、次序以及存储在什么块中。

根据使用方式的不同，数据块分为两种类型：全局数据块和背景数据块。全局数据块我们称之为“自由”数据块，因为它没有被指派给任何代码块；而背景数据块，作为块的局部数据，是与被指定的功能块相关联的。

注意：各种块（除组织块外）的数目和代码的长度是与 CPU 不相关的，而组织块的数目则与 CPU 的操作系统相关。

### 4.2.2 系统块

系统块包含在操作系统中，包括：系统功能（SFC）、系统功能块（SFB）和系统数据块（SDB）。

系统块中包含重要的系统功能函数，如通信功能、操纵 CPU 的内部时钟等。

用户可以调用系统功能和系统功能块，但没有修改的权利。在用户的存储区中，这两个块本身不占据程序空间，而系统功能块（SFB）被调用时，其背景数据块占用用户的存储空间。

操作系统如何调用上述块呢？如图 4-3 所示为块的调用关系。图中上半部分 FB 和 FC 的调用有一定的嵌套关系。

在系统起动过程中，CPU 如何动态工作呢？如图 4-4 所示为 CPU 动态扫描过程。首先，系统上电，开始运行初始化程序 OB100，之后进入可编程的工作周期：进行过程映像输入，运行主程序，再进行过程映像输出。

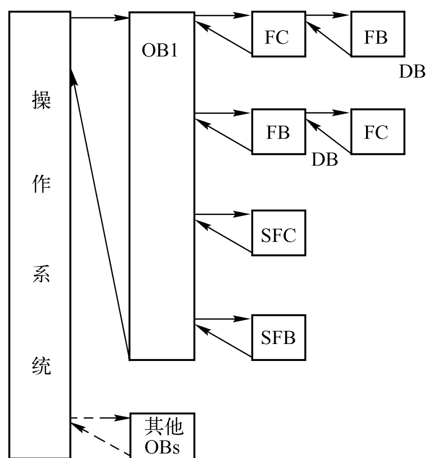



图 4-3 块的调用关系

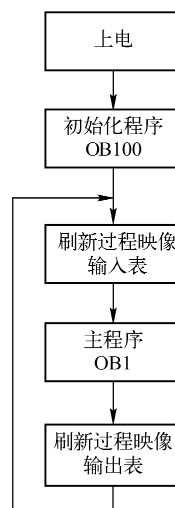



图 4-4 CPU 动态扫描过程

## 4.3　指令结构

### 4.3.1　指令组成

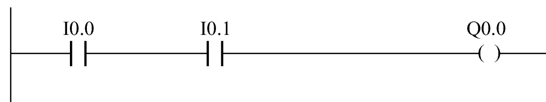
用户程序是由若干条指令顺序排列而成的，而指令是程序的最小单元。指令又是如何构成的呢？我们分别用梯形图和语句表来说明。

在语句表中，一条语句是由操作码和操作数组成的。如一条位逻辑操作指令：

A I0.0

其中，“A”是操作码，表示执行“与”操作；“I0.0”是操作数，表示对输入地址为0的第0位：“I0.0”进行操作。

在梯形图中，指令是由图形元素组成的，其操作码是由图素表示的。例如：




表示对输入“I0.0”和“I0.1”进行“与”操作，并赋值给线圈“Q0.0”。

### 4.3.2　数据类型及存储区

上面提到了“操作数”的概念，因此首先必须对操作数的数据类型作进一步了解。STEP 7 中有三种数据类型：基本数据类型、复合数据类型及参数类型。

**1. 基本数据类型**

对“操作数”进行操作时，是对变量的绝对地址进行操作。而为了程序的可读性，我们对绝对地址进行加工，对应为符号地址。

绝对地址使用的是数字地址，而符号地址使用的是用户所定义的字母名称。它可以是符号表中的全局地址或者是块中声明的局部地址。

（1）变量的绝对地址

变量的绝对地址是在 STEP 7 中硬件组态时设置的输入、输出地址。这里要区分数字信号和模拟信号的不同。

① 数字信号。数字信号包含位信息。如限位开关、点动开关等数字输入信号，以及指示灯、交流接触器等数字输出信号。

② 模拟信号。模拟信号包含16位或32位信息，在可编程序控制器中是以字（WORD）或双字（DWORD）的形式体现的。

数字信号是以布尔（BOOL）量存储的，而模拟信号则以整数（INT）类型来存储。在 STEP 7 中，有以下几种基本数字形式。

1）位（Bit）。变量的数据类型 BOOL 量是通过一个变量标识符、一个地址位、一个分界点和一个字节位来表示的。地址位的上限由 CPU 决定，而字节位范围为0～7。例如：

I1.0 表示输入地址1的第0位；

Q16.4 表示输出地址16的第4位。

2）字节（BYTE）。变量的数据类型 BYTE 是通过地址标识符 B 和表示绝对地址的一个字节来表示的。例如：

IB 2 表示输入地址 2 的字节；

QB 18 表示输出地址 18 的字节。

3）字（WORD）。变量的数据类型 WORD 包含两个字节，同样是通过地址标识符 W 和表示绝对地址的变量高字节所在的地址来表示的。为了避免地址交叉，地址一般为 2 的倍数。例如：

IW 4 表示输入地址是 4，而且包含 IB4 和 IB5 两个字节；

QW 20 表示输出地址 20，而且包含 QB20 和 QB21 两个字节。

字的取值范围为 W#16#0000 ~ W#16#FFFF。

4）双字（DWORD）。变量的数据类型 DWORD 包含四个字节，是通过地址标识符 D 和表示绝对地址的变量高字节所在的地址来表示的。为避免地址交叉，地址一般为 4 的倍数。例如：

ID 8 表示输入地址是 8，而且包含 IB8、IB9、IB10 和 IB11 四个字节；

QD 24 表示输出地址是 24，而且包含 QB24、QB25、QB26 和 QB27 四个字节，如图 4-5 所示为字节、字、双字的排列顺序。双字的取值范围为 W#16#0000_0000 ~ W#16#FFFF_FFFF。

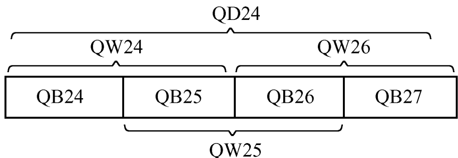



图 4-5　字节、字、双字的排列顺序

5）16 位整数（INT）。整数是有符号数，在进行数字运算时使用，其应用与计算机中的定义相同。最高位为 0 时为正数，为 1 时为负数。取值范围为 $-2^{15} \sim +2^{15}-1$。

6）32 位整数（DINT）。取值范围为 $-2^{31} \sim +2^{31}-1$。

7）32 位浮点数（REAL）。定义同计算机中的格式。这里需注意：PLC 输入输出值大多为整数，若用浮点数处理这些数据时需要进行整数和浮点数之间的转换。

（2）变量的符号地址

符号地址使用符号来取代绝对地址，用户可以根据变量的功能定义方便阅读的符号，而且，符号的定义需要以字母开头，不能用关键字。

根据应用场合的不同，符号分为全局符号和局部符号。

① 全局符号。在下列场合，用户可以定义全局符号。

- 输入 I（输入过程映像存储区）、输出 Q（输出过程映像存储区）、外设输入 PI、外设输出 PQ。
- 位存储区 M、定时器 T、计数器 C。
- 数据块 DB 和代码块 OB、FB、FC。

注意：程序中全局符号与绝对地址对应，所以要唯一。在程序中符号显示在双引号内。在 STEP 7 的菜单中，可以选择以符号的形式显示程序，或以绝对地址的形式显示程序。

② 局部符号。局部符号是归属于功能块的。这些符号的定义只能包含字母、数字和下划线。

局部符号只有在定义的块中才有效。同样的符号可以定义并使用在其他块中，但意义可以完全不同。在程序中局部符号显示在“#”号之后。

（3）常数

我们可以预先给变量设置一个常数值，根据数据类型的不同，常数有不同的前缀。

数据类型规定了数据的特性、允许的范围，如表 4-1 所示显示了基本数据类型。这些数据类型的变量如 BOOL 量，可以进行二进制的逻辑操作、Set/Reset 的功能操作，除 BOOL 量外的其余变量类型可以进行装载和传输指令操作。

**表 4-1　基本数据类型**

| 数据类型 | 描述 | | 常用符号举例 |
|---|---|---|---|
| BOOL | 位 | 1 位 | TRUE，FALSE |
| BYTE | 字节<br>8 位十六进制数 | 8 位 | B#16#00（最小值）<br>B#16#FF（最大值） |
| CHAR | 字符（ASCII） | 8 位 | ‘A’ |
| WORD | 字<br>16 位十六进制数<br><br>16 位二进制数<br><br>计数器值 3 位 BCD 码<br>2 个 8 位无符号十进制数 | 16 位 | W#16#0000（最小值）<br>W#16#FFFF（最大值）<br>2#0000_0000_0000_0000<br>2#1111_1111_1111_1111<br>C#000（最小值）<br>C#999（最大值）<br>B（0，0）（最小值）<br>B（255，255）（最大值） |
| DWORD | 双字<br>32 位十六进制数<br>4 个 8 位无符号十进制数 | 32 位 | DW#16#0000_0000（最小值）<br>DW#16#FFFF_FFFF（最大值）<br>B（0，0，0，0）（最小值）<br>B（255，255，255，255）（最大值） |
| INT | 定点数 | 16 位 | -32768（最小值）<br>+32767（最大值） |
| DINT | 定点数 | 32 位 | -2 147 483 648（最小值）<br>+2 147 483 647（最大值） |
| REAL | 浮点数 | 32 位 | + 123.4567 具有小数的十进制数或 1.234567E+02 指数形式表示 |
| S5TIME | S5 格式时间值 | 16 位 | S5T#0 ms（最小值）<br>S5TIME#2 h46 m30 s（最大值） |
| TIME | IEC 格式时间值 | 32 位 | T# -24d20h31 m23 s647 ms　（最小值）<br>TIME#24d20h31 m23 s647 ms　（最大值） |
| DATE | 日期 | 16 位 | D#1990_01_01（最小值）<br>Date#2089_12_31（最大值） |
| TIME_OF_DAY | 时间日期 | 32 位 | TOD#00：00：00：000（最小值）<br>TIME_OF_DAY#23：59：59.999（最大值） |

## 2. 复合数据类型

复合数据有以下几种类型：

① 数组（ARRAY）：将同一类型的数据合成一组，形成一个单元。在数据块中常使用这种类型。

② 结构（STRUCT）：将不同类型的数据合成一组，形成一个单元。

③ 字符串（STRING）：将多个字符（CHAR）组成一维数组，形成字符串。

其余还有日期和时间类型，用户定义的数据类型 UDT。其中 UDT 类型在 FB 块中也常常使用。

**3．参数类型**

参数类型是为逻辑块之间传递形参而设定的。形式如下：

① TIMER 和 COUNT：如 T3、C21。

② BLOCK：如 FB、FC。

③ POINT：如 P#M0.0 表示指向 M0.0 的指针。

④ ANY：用于实参的数据类型。如 P#DB1.DBX0.0 BYTE 30 表示 DB1 中以 0 地址为起始地址的 30 个字节。在数据通信中经常使用。

### 4.3.3　CPU 存储区

存储区及功能如表 4-2 所示，其中所述的最大地址范围是由 CPU 型号和硬件配置来决定的，而表中给出的地址范围是一般的规定。

注意：与直接访问外设 I/O 相比，访问过程映像表可以保证在一个程序周期内过程映像的状态始终一致。如若在程序执行过程中输入模块外部信号发生变化，则过程映像值直到下一个循环周期才被刷新。这是 PLC 顺序循环扫描的特点。

### 4.3.4　寻址方式

既然前面提到操作数，那么就一定要有相应的寻址方式，S7 中有四种寻址方式，分别为立即寻址、存储器直接寻址、存储器间接寻址、寄存器间接寻址。在语句表编程中寻址方式的使用十分频繁。

**1．立即寻址**

立即寻址是操作数为常见的常数或常量的寻址方式。其操作数直接包含在指令中。立即寻址的具体应用如下：

```
SET          //置逻辑操作结果为 1
L  +27       //将整数 27 装入累加器 1 中
L C#09       //将 BCD 码常数 9 装入累加器 1 中
L P#10.0     //将内部区域指针装入累加器 1 中
```

**2．直接寻址**

直接寻址是对寄存器和存储器进行的直接寻址方式。它直接给出寄存器或存储器的地址。直接寻址的具体应用如下：

```
A I0.1          //对输入位 I0.1 进行“与”操作
S Q0.1          //将输出位 Q0.1 置 1
L MW4           //将 MW4 值装入累加器 1 中
```

T MW10　　　　　　//把累加器 1 中低字数值传给 MW10 中

**表 4-2　存储区及功能**

| 区域 | 区域功能 | 访问区域单位 | 标识符 | 最大地址范围 |
|---|---|---|---|---|
| 输入过程映像存储区（I） | 在循环扫描开始时，从过程中读取输入信号至过程映像存储区 | 输入位<br>输入字节<br>输入字<br>输入双字 | I<br>IB<br>IW<br>ID | 0～65 535.7<br>0～65 535<br>0～65 534<br>0～65 532 |
| 输出过程映像存储区（Q） | 在循环扫描期间，将过程映像存储区中的输出值传至输出模块 | 输出位<br>输出字节<br>输出字<br>输出双字 | Q<br>QB<br>QW<br>QD | 0～65 535.7<br>0～65 535<br>0～65 534<br>0～65 532 |
| 位存储区（M） | 此存储区用于存储控制逻辑的中间状态 | 存储器位<br>存储器字节<br>存储器字<br>存储器双字 | M<br>MB<br>MW<br>MD | 0～255.7<br>0～255<br>0～254<br>0～252 |
| 外部输入（PI）<br>外部输出（PQ） | 用户可通过此区域直接访问输入和输出模块 | 外部输入字节<br>外部输入字<br>外部输入双字<br>外部输出字节<br>外部输出字<br>外部输出双字 | PIB<br>PIW<br>PID<br>PQB<br>PQW<br>PQD | 0～65 535<br>0～65 534<br>0～65 532<br>0～65 535<br>0～65 534<br>0～65 532 |
| 定时器（T） | 访问此区域可以得到定时剩余时间 | 定时器（T） | T | 0～255 |
| 计数器（C） | 访问此区域可以得到当前计数值 | 计数器（C） | C | 0～255 |
| 数据块（DB） | 用“OPEN DB”打开数据块，用“OPEN DI”打开背景数据块 | 数据位<br>数据字节<br>数据字<br>数据双字 | DB（I）X<br>DB（I）B<br>DB（I）W<br>DB（I）D | 0～65 535.7<br>0～65 535<br>0～65 534<br>0～65 532 |
| 本地数据（L） | 此区域存放逻辑块中的临时数据，当逻辑块结束时，数据丢失 | 临时本地数据位<br>临时本地数据字节<br>临时本地数据字<br>临时本地数据双字 | L<br>LB<br>LW<br>LD | 0～65 535.7<br>0～65 535<br>0～65 534<br>0～65 532 |

### 3. 存储器间接寻址

在存储器间接寻址指令中，给出一个作地址指针的存储器，存储器的内容是操作数所在存储单元的地址。地址指针可以是字，也可以是双字。其中定时器、计数器、数据块、功能块用字指针。存储器间接寻址中双字指针的格式如图 4-6 所示。其中 0～2 位为被寻址地址中位的编号，3～18 为寻址字节编号，且只有双字 MD、LD、DBD 和 DID 才能作地址指针。

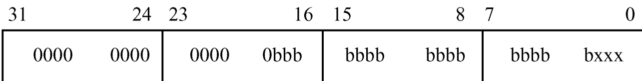



图 4-6　存储器间接寻址中双字指针的格式

如果用双字格式的指针访问一个字、字节或双字存储器，需保证指针的位编号为零。存储器间接寻址的具体应用如下：

```
L QB[DBD10]          //若 DBD10 的值为 2#0000 0000 0000 0000 0000 0000 0010 0000,则装载
                       QB4 内容
L P#8.7              //装载指针值到累加器中
T MD2                //传输指针到 MD2 中
A I[MD2]             //扫描输入状态位 I 8.7
= Q[MD2]             //将其赋值给 Q 8.7
```

存储器间接寻址的优点是：当程序执行时，可以改变操作数的存储地址。这在循环编程中十分重要。

4．寄存器间接寻址

AR1 和 AR2 是 S7 中的两个地址寄存器，通过它们可以对各存储器内容作寄存器间接寻址。地址寄存器的内容作为基址，加上偏移量，指向其存储单元。

寄存器间接寻址中双字地址指针格式如图 4-7 所示。

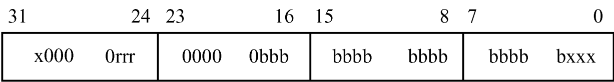



图 4-7　寄存器间接寻址中双字地址指针格式

其中 0 ~ 2 位指被寻址地址中位的编号，3 ~ 18 位为被寻址地址的字节编号，24 ~ 26 位为被寻址地址的区域标识符。常用的区域标识符如 M 的区域标识符为 011；I 的区域标识符为 001；Q 的区域标识符为 010；外设输入输出 P 的区域标识符为 000。

有两种地址指针格式：区内寻址和区域间寻址。区内寻址利用双字指针中的字节编号和位编号，存储器的类型在指令中给出；区域间寻址同时还利用了区域标识符（rrr），这样可实现区域间的间接寻址。

如果用寄存器指针访问一个字节、字或双字，必须保证指针中位地址编号为零。寄存器区内寻址的具体应用如下：

```
L   P#8.7              //装载指针值到累加器 1 中
LAR1                   //将累加器 1 中内容送到地址寄存器 1 中
A   I[AR1, P#0.0]      //扫描输入位 I 8.7
=   Q[AR1, P#1.1]      //将值赋给输出位 Q 10.0。地址为 8.7（AR1）加 1.1（偏移量）后为
                         10.0,而不是 9.8。
```

## 4.3.5　状态字和逻辑操作过程

1．状态字

状态字用于表示 CPU 执行指令时所具有的状态。当指令执行时，状态字内容随之变化，并提供程序运行结果及错误信息。在必要情况下，用户可以在位逻辑指令和字逻辑指令中访问并检测状态字，同时采取相应的行动。如图 4-8 所示为状态字的结构。

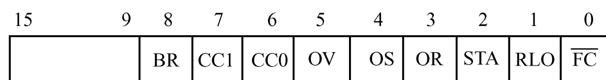



图 4-8　状态字结构

① 首次检测位（$\overline{FC}$）。状态字的第 0 位为首次检测位（First Check）。首次检测位表示逻辑操作的开始状态，它具有特殊的意义，因为 CPU 会直接接受新的 RLO，而旧的 RLO 随时丢弃。

如果$\overline{FC}$值为 0，表示逻辑串的开始；在逻辑串的执行过程中，$\overline{FC}$为 1；当执行输出指令、置位/复位指令或转移指令时，将$\overline{FC}$清零，逻辑串结束。

② 逻辑操作结果（RLO）。状态字的第 1 位为逻辑操作结果（Result of the logic operation），它存储位逻辑指令和算术比较指令的结果。

在程序的第一条指令中，检测信号状态，执行后，将 RLO 置 1；在程序的第二条指令中，检测信号状态，根据逻辑关系，将其与原先存储的 RLO 值组合，得出新的 RLO 值并存储，之后以此类推。当输出指令或转移指令执行时，逻辑串结束。

③ 状态位（STA）。状态字的第 2 位为状态位（Status Bit），顾名思义是表示信号的状态，可以是“0”，也可以是“1”。当电压加在输入端上，状态为“1”，否则为“0”。状态位不能用于指令检测，它只是在程序测试中被 CPU 解释使用。

④ 或位（OR）。状态字的第 3 位为或位（OR Bit），当逻辑操作是 AND 和 OR 的组合时，或状态位（OR）存储 AND 的操作结果，为后续的 OR 操作做准备。当完成组合操作时，重新清零。

⑤ 溢出状态保持位（OS）。状态字的第 5 位为溢出状态保持位（Overflow Stored），当溢出、非法操作发生时，OS 位置 1，同时还具有保存 OV 位的作用：在 OV 被置 1 时，OS 位也被置 1，OV 被清零，OS 位依然保持。

⑥ 溢出位（OV）。状态字的第 4 位为溢出位（Overflow），当一个算术运算或浮点数比较指令执行时出现错误，则溢出位置 1。只有执行结果正常后，溢出位被清零。

⑦ 条件码 1（CC1）和条件码 0（CC0）。状态字的第 6 位和第 7 位为条件码（Condition Codes）0 和条件码 1，在位逻辑指令、比较指令、算术指令、移位和循环移位指令以及字逻辑运算中，条件码都有相应的值表示与零的大小关系。

⑧ 二进制结果位（BR）。状态字的第 8 位为二进制结果位（Binary Result Bit），在梯形图中，BR 位与 ENO（使能输出）有对应关系，用于表示方块指令是否正确执行；如果执行错误，BR 位为 0，ENO 也为 0，反之，两者均为 1。

在用户编写的 FC 和 FB 程序中，必须对 BR 位进行管理。在 STL 指令中，SAVE 指令可以将 RLO 存入 BR 中，从而达到管理 BR 位的目的。当 FC 和 FB 程序执行无错误时，将 RLO 置 1 并存入 BR 中，否则 BR 为 0。

**2. 逻辑操作过程**

有了状态字的概念后，就可以了解逻辑操作过程的全貌。如图 4-9 所示为逻辑操作过程。

1）首先，输入模块选择一个传感器作为输入。

2）CPU 检测这个输入状态，为后续的逻辑操作做准备。

3）逻辑操作结果存入 RLO 中。

4）之后 CPU 对输出模块进行操作。

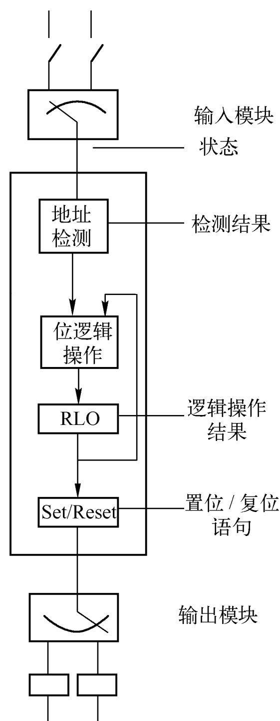



图 4-9　逻辑操作过程

## 4.4　位逻辑指令

位逻辑指令主要包括位逻辑运算指令、位操作指令和位测试指令。它们完成逻辑操作，并将逻辑操作结果 RLO 用于赋值或置位，也用于控制定时器和计数器的运行。

### 4.4.1　位逻辑运算指令

位逻辑运算指令是“与”（AND）、“或”（OR）和“异或”（XOR）及这些指令的组合。

**1．与、或、异或指令的形式**

在语句表中与、或和异或指令用做检测二进制位的信号状态，并将其联系在一起；在梯形图中，是以继电器的串联、并联等形式将二进制信号组合在一起；而在功能块图中，是以框图来模拟电气开关系统的布尔功能。

梯形图和功能块图的基本逻辑如图 4-10 所示。

语句表基本指令为：

A　位地址

检测信号为 1，和 AND 指令相关联，表示串联的常开接点。

AN　位地址

检测信号为 0 ，同样与 AND 指令相关联，表示串联的常闭接点。

| 功能块图逻辑 | | 梯形图逻辑 | |
|---|---|---|---|
| 与 | & | 常开接点 | |
| 或 | >=1 | 常闭接点 | |
| 异或 | XOR | 取反 | NOT |
| 取反 | | | |
| 输出 | = | 线圈 | |

图 4-10　梯形图和功能块图的基本逻辑

O　位地址

检测信号为 1，和 OR 指令相关联，表示并联的常开接点。

ON　位地址

检测信号为 0，和 OR 指令相关联。

X　位地址

检测信号为 1，和 XOR 指令相关联。

XN　位地址

检测信号为 0，和 XOR 指令相关联。

NOT　取反逻辑操作指令

=　位地址

输出指令，将逻辑操作结果 RLO 赋给地址位。

与、或和异或指令在梯形图中和功能块图中的表示如图 4-11a、b 所示。

说明：图 4-11a 中，当常开接点 I0.0 与常闭接点 I0.1 接通时，输出 Q4.0 得电。I0.0 和 I0.1 为串联关系；当常开接点 I0.2 和 I0.3 中有一个接通时，输出 Q4.1 得电。I0.2 和 I0.3 为并联关系；当常开接点 I0.3 和 I0.4 扫描状态不同，即只有一个为“1”时，逻辑操作结果 RLO 为“1”，并赋值使输出 Q4.2 得电；若 I0.3 和 I0.4 扫描状态相同时，RLO 为“0”，输出 Q4.2 断电，I0.3 和 I0.4 为异或关系。

图 4-11b 中，功能块图的串联、并联及异或关系比较明确，这里就不再解释了。

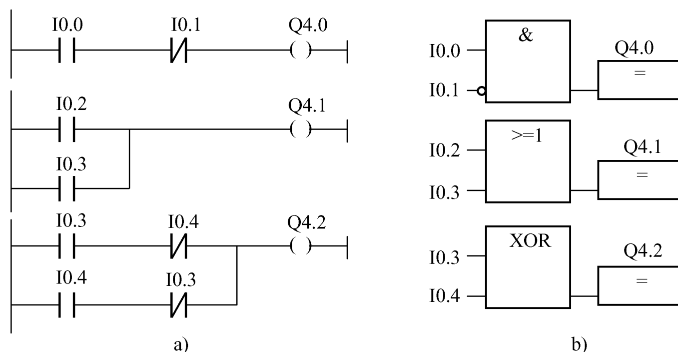



图 4-11　与、或和异或指令在梯形图、功能图中的表示

与、或和异或指令的语句表形式如下：

```
A    I0.0
AN        I0.1 //I0.0 与
 =        Q4.0 //RLO 结果赋值给 Q4.0

O         I0.2
O         I0.3 //I0.2 或 I0.3
 =        Q4.1 //RLO 结果赋值给 Q4.1

X         I0.3
X         I0.4 //I0.3 异或 I0.4
 =        Q4.2 //RLO 结果赋值给 Q4.2
```

使用场合：在这些指令中，“与”指令非常有用。这是因为 PLC 中不具备掩码功能。所谓掩码是指当用户需要寄存器的某些位为“1”工作，而某些位为“0”不工作，这些可以在机器中自行设置。PLC 既然不具备这样的功能，我们可以利用“与”指令来实现之。用户只需将其中需要保留的位和“1”与，不需保留的位和“0”与即可。怎么样，有了“与”指令，事情变得简单了。比如 2#0100_0010 中第 1 位需要保留，“与”2#0000_0010 就可以实现。

“异或”指令就是我们常说的相同出“0”，不同出“1”的概念。当用户要比较两个寄存器内容时，或者在通信中进行奇偶校验时，使用异或指令将十分便捷。

**例 1**　设计电动机的正反转，其中正转点动按钮为 I0.0，反转点动按钮为 I0.1，停止按钮为 I0.2，正转线圈为 Q0.0，反转线圈为 Q0.1。要求：正转和反转线圈要有“互锁”，正、反转按钮有“互锁”，正转点动按钮要“自锁”，反转点动按钮同样如此要求。

电动机的正反转设计是最简单的 PLC 程序，如图 4-12 所示。

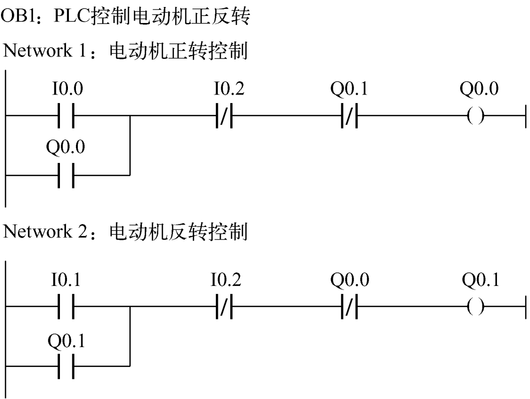



图 4-12　电动机正反转 PLC 梯形图程序

说明：在 Network1 和 Network2 中通过“与”相应的常闭接点实现正反转线圈之间的互锁。通过“或”指令实现了正转和反转线圈各自的自锁。

前面已经介绍了各个状态位的含义。在学习了基本位逻辑运算指令后，通过表 4-3 和表 4-4 我们可以加深了解在程序执行中状态位的变化情况。表 4-3 和表 4-4 为状态位的变化表。

**表 4-3　状态位的变化（注意 STA 和 OR 的变化）**

| STL 语句 | STA | RLO | /FC | OR | 注释 | |
|---|---|---|---|---|---|---|
| AN | I0. 0 | 0 | 1 | 1 | 0 | I0. 0 为“0” |
| A | I0. 1 | 1 | 1 | 1 | 0 | I0. 1 为“1” |
| A | I0. 2 | 1 | 1 | 1 | 0 | I0. 2 为“1” |
| O | | 1 | 1 | 1 | 1 | |
| A | I0. 3 | 1 | 1 | 1 | 1 | I0. 3 为“1” |
| A | I0. 4 | 1 | 1 | 1 | 1 | I0. 4 为“1” |
| O | I0. 5 | 1 | 1 | 1 | 0 | I0. 5 为“1” |
| = | Q4. 0 | 1 | 1 | 0 | 0 | Q4. 0 置为“1” |
| R | Q4. 1 | 0 | 1 | 0 | 0 | Q4. 1 置为“0” |
| S | Q4. 2 | 1 | 1 | 0 | 0 | Q4. 2 置为“1” |

**表 4-4　状态位的变化（注意 STA、/FC 和 RLO 的变化）**

| STL 语句 | | STA | Check Result | RLO | /FC | OR | 注释 |
|---|---|---|---|---|---|---|---|
| = | M1. 0 | X | | X | 0 | | |
| A | I0. 0 | 1 | 1 | 1 | 1 | 0 | I0. 0 为“1” |
| A | I0. 1 | 1 | 1 | 1 | 1 | 0 | I0. 1 为“1” |
| AN | I0. 2 | 0 | 1 | 1 | 1 | 0 | I0. 2 为“0” |
| = | Q4. 0 | 1 | | 1 | 0 | 0 | Q4. 0 置为“1” |
| R | Q4. 1 | 0 | | 1 | 0 | 0 | Q4. 1 置为“0” |
| S | Q4. 2 | 1 | | 1 | 0 | 0 | Q4. 2 置为“1” |
| ON | I1. 0 | 0 | 1 | 1 | 1 | 0 | I1. 0 为“0” |
| O | I1. 1 | 1 | 1 | 1 | 1 | 0 | I1. 1 为“1” |
| O | I1. 2 | 1 | 1 | 1 | 1 | 0 | I1. 2 为“1” |
| = | Q5. 0 | 1 | | 1 | 0 | 0 | Q5. 0 置为“1” |
| R | Q5. 1 | 0 | | 1 | 0 | 0 | Q5. 1 置为“0” |
| S | Q5. 2 | 1 | | 1 | 0 | 0 | Q5. 2 置为“1” |

说明：从表 4-3 中可以看出：STA 的状态与信号状态一致。如 I0. 0 的信号状态为“0”，所以 STA 状态为“0”，I0. 1 的信号状态为“1”，所以 STA 状态为“1”；由于与、或指令交叉进行，所以状态位 OR 存储 AND 的操作结果，表现在表的第 4 行，OR 状态为“1”，为后续的 OR 操作做准备。当完成组合操作时，重新清零，表现在表的第 7 行，OR 状态为“0”。

以同样的方式，用户可以观察表 4-4 中 STA、/FC 和 RLO 的变化。

### 2. 位逻辑指令的组合

位逻辑指令可以任意组合，不过，组合有优先权。用户当然知道乘除法在加、减法之先，AND 相当于乘法，优先权在 OR 和 XOR 之前。而 OR 和 XOR 在 STL 语句中具有相同的优先权。

STL 语句中的“括弧”的作用同四则运算中的“括弧”的作用。为了能按正确的功能顺序进行工作，可以加“括弧”，使 CPU 暂时在 RLO 中存储临时结果，“括弧”使之具有优先权。STL 语句加括弧前后在功能块图中的表现如表 4-5 所示。

**表 4-5　STL 语句加括弧前后在功能块图中的表现**

| | |
|---|---|
| input0, input1, input2 → & ; input3, input4 → & ; input5 → >=1 → Output | AN Input0<br>A Input1<br>A Input2<br>O<br>A Input3<br>A Input4<br>O Input5<br>= Output |

（续）

| | | |
|---|---|---|
| input0 input1 input2 >=1; input3 input4 >=1; input5; & Output | A (<br>O Input0<br>O Input1<br>O Input2<br>)<br>A (<br>ON Input3<br>ON Input4<br>)<br>A Input5<br>= Output | O Input0<br>O Input1<br>O Input2<br>A (<br>ON Input3<br>ON Input4<br>)<br>A Input5<br>= Output |

说明：在表的第 1 行中，由于语句表中与指令比或指令有优先权，所以表现在功能块图中为先与后或；在表的第 2 行中，由于“括弧”将或指令包括在内，其优先权比与指令高，所以表现在功能块图中为先或后与。

## 4.4.2 位操作指令

### 1. 赋值指令（输出指令）

赋值指令（=）将逻辑操作结果 RLO 写入指定地址位。

在梯形图中，输出指令只能放在逻辑符号串的最右端，且前面必须有常闭或常开接点。

一个 RLO 可以驱动几个输出元件，输出线圈从上至下排列。如图 4-13 所示为梯形图和语句表的输出指令。对应 STL 中，输出具有相同的优先级。

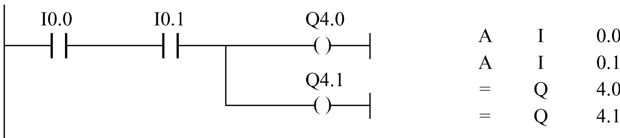



图 4-13　梯形图和语句表的输出指令

### 2. RS 触发指令

在正反转电路中，如果没有正反转线圈的自锁，那么，正转或反转按钮必须一直按着，才能保证电动机正转或者反转。这显然太麻烦，能否像电视遥控器的数字按钮那样，按动数字键 1 后，频道 1 开，按动其他按钮，可切换到其他频道呢？PLC 中有此功能的指令是 RS 触发器。

RS 触发指令在功能块图及梯形图中的表现如图 4-14 所示，其中有触发器线圈和触发器方块两种形式。

语句表中，置位和复位语句表示为：

```
S   Bit     //在 RLO 为 1 时,将 Bit 置 1
R   Bit     //在 RLO 为 1 时,将 Bit 清零
```

注意：置位 S 和复位 R 在编程中常常成对出现，并一一对应。

置位和复位线圈指令在梯形图和功能块图中的具体应用如图 4-15a、b 所示。

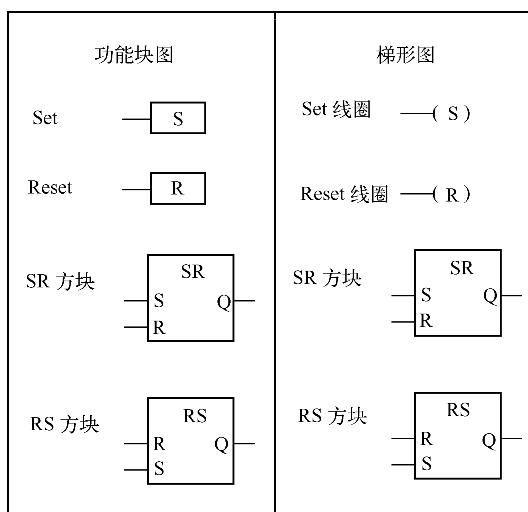



图 4-14　RS 触发指令在功能块图及梯形图中的表现

说明：当接点 I0. 0 接通时，置位 Q4. 0；当 I0. 1 接通时，复位 Q4. 0。由于 PLC 顺序扫描，所以当 I0. 0 和 I0. 1 同时为“1”时，为复位优先，Q4. 0 复位。

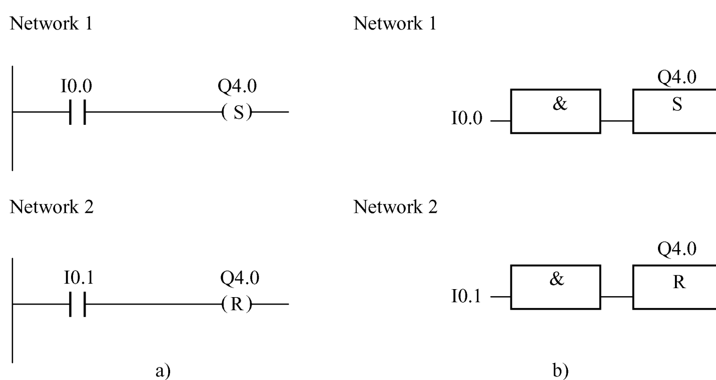



图 4-15　置位和复位线圈指令在梯形图和功能块图中的具体应用

实现上述相同功能的语句表程序为：

```
A   I0. 0
S   Q4. 0          //当 I0. 0 为“1”,置位 Q4. 0
A   I0. 1
R   Q4. 0          //当 I0. 1 为“1”,复位 Q4. 0
```

在梯形图和功能块图中 RS 触发器有置位优先和复位优先两种类型，在置位优先型 RS 触发器中，R 端在 S 端之上，当两个输入端都为 1 时，置位输入最终有效；复位优先型反之，S 端在 R 端之上，当两个输入端都为 1 时，复位输入最终有效。这是因为 CPU 顺序扫描的结果。复位优先的 RS 触发器在梯形图和功能块图中的具体应用如图 4-16a、b 所示。

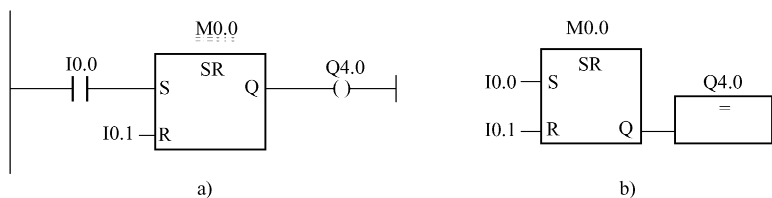



图 4-16　复位优先的 RS 触发器在梯形图和功能块图中的具体应用

说明：I0.0 接通时，置位 Q4.0；I0.1 接通时，复位 Q4.0，且复位优先。中间结果存于存储位 M0.0 中，中间存储位的状态与输出是相同的。

注意：当程序中有多条置位复位触发器时，注意中间存储位不能重复。

实现上述相同功能的语句表程序为：

```
A   I0.0
S   M0.0        //置位 M0.0,存储中间结果
A   I0.1
R   M0.0        //复位 M0.0
A   M0.0
=   Q4.0        //将 M0.0 结果赋值给 Q4.0
```

注意：语句表中若置位指令在先，复位指令在后，则实现复位优先触发器的功能；反之则实现置位优先触发器的功能。

**例 2**　用 RS 触发器设计电动机正反转电路程序。要求同前。

用 RS 触发器实现正反转电路程序如图 4-17 所示。

说明：程序用触发器实现会变得比较简单，这里我们用了复位优先型触发器，因为一般情况下，在故障出现后，紧急停车是当务之急。这样当 I0.2 按钮按下时，复位优先，电动机将停止运行。由于触发器置位的功能，程序中的输入按钮 I0.0（正转按钮）和 I0.1（反转按钮）按下时，输出不需要自锁，而输入按钮和正反转输出线圈的互锁功能依然同前。

注意：由于中间存储位与输出状态相同，所以省略了输出线圈。

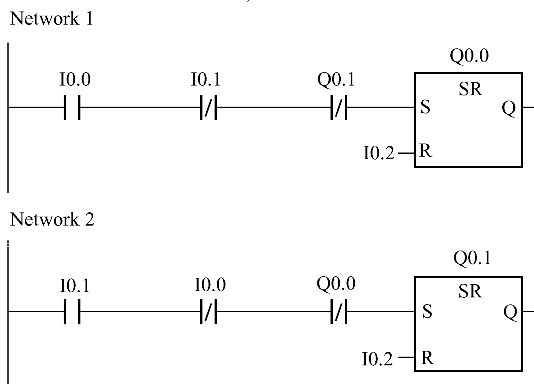



图 4-17　用 RS 触发器实现正反转电路程序

**例 3**　抢答器的设计：抢答器有三个输入，分别为 I0.0、I0.1 和 I0.2，输出分别为 Q4.0、Q4.1 和 Q4.2，复位输入是 I0.4。要求：三人中任意抢答，谁先按按钮，谁的指示灯优先亮，且只能亮一盏灯，进行下一问题时主持人按复位按钮，抢答重新开始。

抢答器的设计程序如图 4-18 所示。

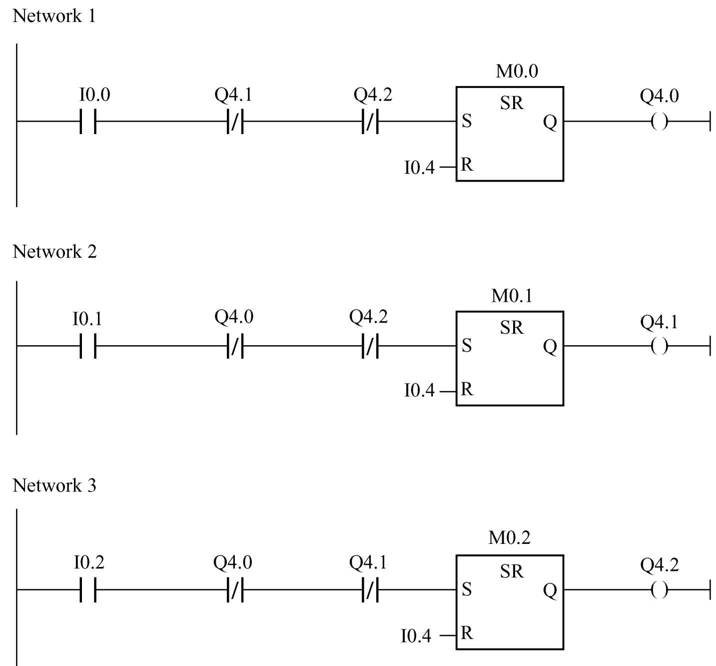



图 4-18　抢答器的设计程序

说明：用复位优先型触发器，用 I0.0、I0.1 和 I0.2 分别置位 Q4.0、Q4.1 和 Q4.2，并且用 I0.4 复位输出。程序中必须有三个输出 Q4.0、Q4.1、Q4.2 的互锁，否则，就不止一盏指示灯亮。中间存储位为 M0.0、M0.1 和 M0.2，存储位各不相同，保证程序正确执行。

注意：触发器的使用中需注意选择触发器的类型，一般选择复位优先型触发器。由于置位和复位的特点，点动按钮的编程（不需自锁）变得相对简单了。

**例 4**　七段码显示如图 4-19 所示。当输入 I0.0 为 1 时，要求七段码输出显示为 0，当 I0.1 为 1 时，显示为 1，试编程实现。

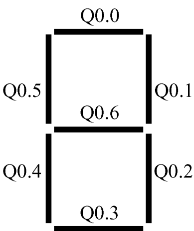



图 4-19　七段码显示

如果用输出线圈指令实现，习惯性的程序编写如图 4-20 所示。

说明：程序存在的问题是没有考虑到 PLC 循环扫描的特点。输出线圈 Q0.1 和 Q0.2 都分别在两段程序中出现，当程序的多处都出现同一输出线圈时，我们称之为多线圈问题。本例中直接导致输入为 I0.0 时，显示的 0 值不对。因为 I0.1 此时为 0，输出 Q0.1 和 Q0.2 不显示。后面的程序刷新了前面的程序。正确的编程如图 4-21 所示。

程序中采用并联的方式，避免了多线圈的出现。解决方法归纳为：当多线圈出现时，将多线圈的输入全部并联。

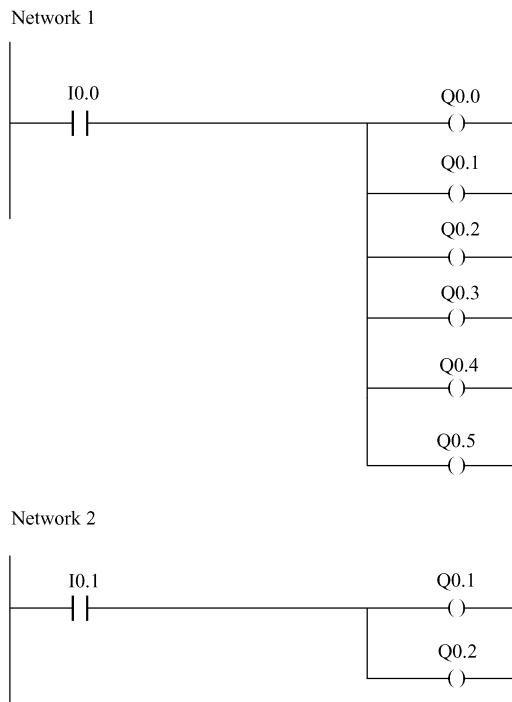



图 4-20　七段码习惯编写程序

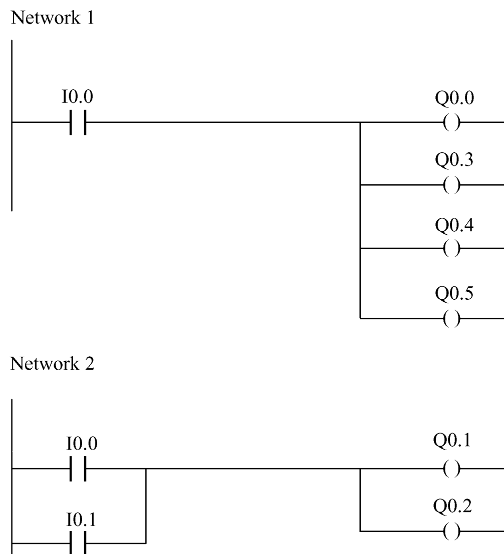



图 4-21　七段码正确的编写程序

程序中采用并联的方式，避免了多线圈的出现。解决方法归纳为：当多线圈出现时，将多线圈的输入全部并联。

如果我们用置位和复位语句又该如何编程呢？程序如图 4-22 所示。

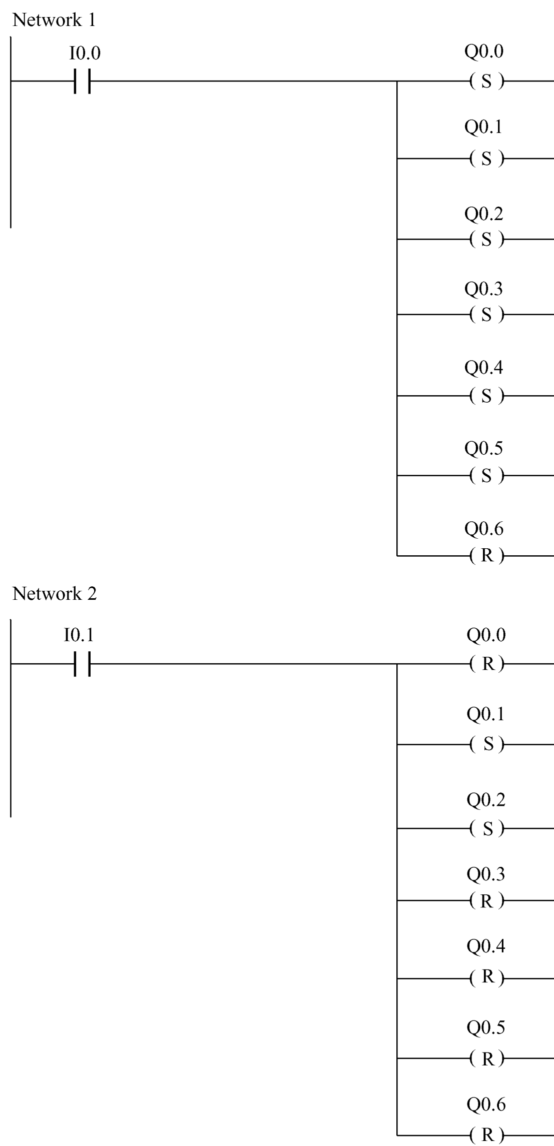



图 4-22　用置位和复位语句编写的七段码程序

说明：程序在显示 1 时，需要那么多的复位语句吗？答案是需要。比如显示了 8 之后，再想显示 1，如果没有复位语句，结果就不会出现 1，还是显示 8。因为其他不该显示的线圈状态还是为 1。置位和复位语句使用时一般都成对出现。有对某个值置位的语句，必然存在对某个值的复位语句。

**3. RLO 边沿检测指令**

边沿检测指令（梯形图和功能块图）如图 4-23 所示。边沿正跳沿指令捕捉到正跳沿后，产生一个扫描周期宽度的脉冲；而边沿负跳沿指令捕捉到负跳沿后，产生一个扫描周期宽度的脉冲，边沿检测波形图如图 4-24 所示。边沿检测常用于只扫描一次的情况，比如，在程序开始，我们给一个变量赋了初值，如果不加边沿检测指令，由于 PLC 顺序循环扫描

的特点，这个变量将永远是初始值，而不发生任何变化。

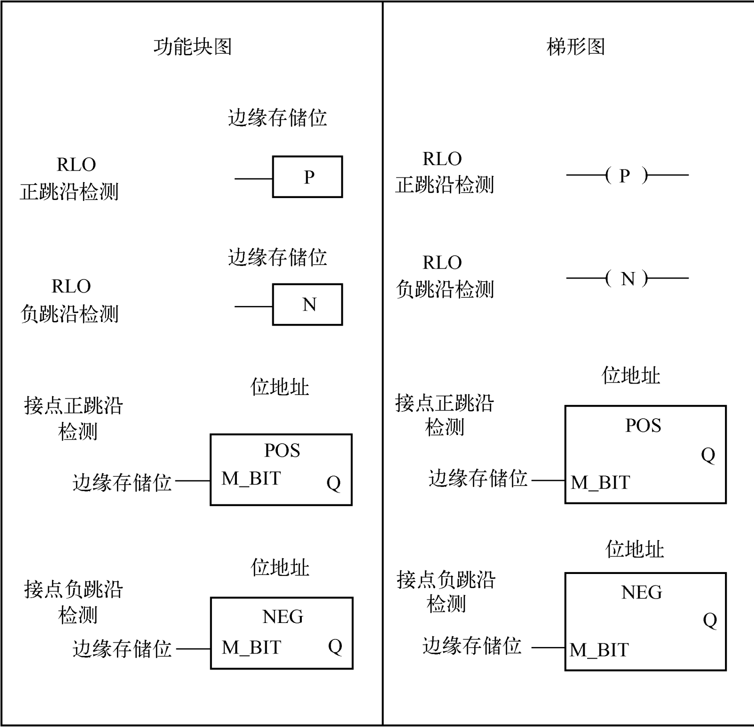



图 4-23　边沿检测指令（梯形图和功能块图）

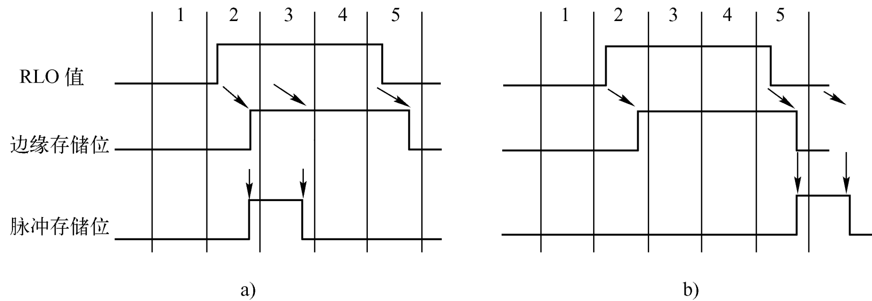



图 4-24　边沿检测波形图

语句表中，正边沿触发和负边沿触发指令表示为：

FP　存储 Bit　//在 RLO 的正跳沿,将存储位在一个扫描周期内置 1

FN　存储 Bit　//在 RLO 的负跳沿,将存储位在一个扫描周期内置 1

正跳沿指令在梯形图和功能块图中的具体应用如图 4-25a、b 所示。

说明：当 I0.0 接通时，RLO 的正跳沿使输出 Q4.0 接通一个扫描周期，中间存储位为 M0.0。

实现上述相同功能的语句表程序为：

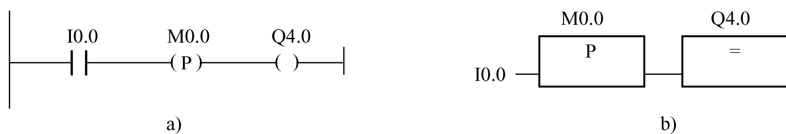



图 4-25　正跳沿指令在梯形图和功能块图中的具体应用

```
A   I0.0
FP   M0.0      //在 RLO 的正跳沿,在 M0.0 中存储一个扫描周期的高电平
 =   Q4.0
```

边沿检测指令十分有用，开始用户会觉得难于理解，不知如何去使用它，我们举例来帮助理解它的工作过程。

**例 5**　设计一个乒乓电路，按动按钮 I0.0，使灯泡亮，再按按钮，灯泡灭。

乒乓电路的梯形图如图 4-26 所示。

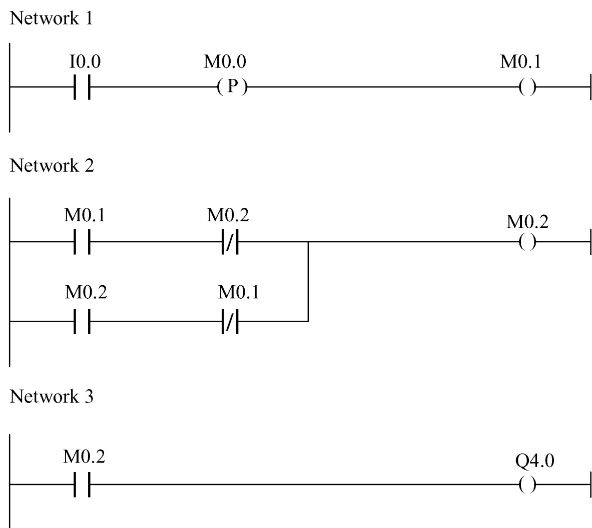



图 4-26　乒乓电路的梯形图

说明：我们分三步解释扫描过程：

第一步：在 Network1 中，第一次按动按钮时，I0.0 接通，在一个扫描周期中，则 M0.1 通；Network2 的第一分支中，由于 M0.1 通，而 M0.2 常闭接点本身是通的，所以 M0.2 接通；在 Network3 中，M0.2 通，所以 Q4.0 通，电灯亮。

第二步：在 Network1 中，由于扫描周期已过，所以，虽然 I0.0 保持不变，M0.1 断开；Network2 中的第一分支，用户仔细分析会发现能流不通，而第二分支中，由于 M0.2 是通的，同时常闭接点 M0.1 是通的，所以，M0.2 保持接通状态。在 Network3 中，由于 M0.2 保持接通，所以灯泡依然亮。

第三步：刚才实现 I0.0 点按一次，电灯亮的过程。之后，当 I0.0 再一次点按时，在 Network1 中，M0.1 接通；在 Network2 中，由于 M0.1 和 M0.2 是通的，所以第一分支和第

二分支能流不通，因而，M0.2 变为不通状态；在 Network3 中，由于 M0.2 不通，所以 Q4.0 不通，电灯灭，之后系统循环运行。

**4. 对 RLO 的直接操作指令**

这类指令可直接对逻辑操作结果 RLO 进行操作，改变状态字中 RLO 的状态。如表 4-6 所示为对 RLO 直接操作的指令。

**表 4-6 对 RLO 直接操作的指令**

| 梯形图指令 | 功能图指令 | STL 指令 | 功能 | 说明 |
|---|---|---|---|---|
| -｜NOT｜- | —o\| | NOT | 取反 RLO | 在逻辑串中，对当前 RLO 取反 |
| | | SET | 置位 RLO | 将 RLO 置 1 |
| | | CLR | 复位 RLO | 将 RLO 清零 |
| -（SAVE） | | SAVE | 保存 RLO | 将 RLO 存入状态字的 BR 位 |

## 习题 I

1. 如图 4-27 所示设计开关电路，使线圈 Q4.0 通电、断电。

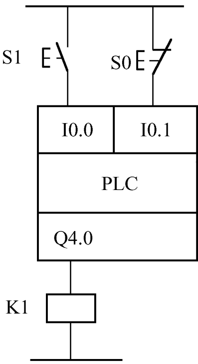



图 4-27 习题 1 图

2. 完成如图 4-28 所示的时序，了解置位和复位语句的用法。STL 程序为：

```
A   I1
A   I2
S   Q1
=Q2
O   I3
O   I4
R   Q1
```

I1
I2
I3
I4
Q1
Q2

图 4-28　习题 2 图

3. 完成如图 4-29 所示的时序，了解边沿触发语句的用法。功能块图如图 4-29 所示。



图 4-29　习题 3 图

4. 将如图 4-30 所示的功能块图转换为 LAD 和 STL 形式。

5. 单按钮控制的要求是只用一个按钮就能控制一台电动机的起动和停止。控制任务是按一次按钮电动机起动，再按一次按钮，电动机停止。试利用置位和复位语句实现控制。



图 4-30　习题 4 图

# 4.5　定时器与计数器指令

## 4.5.1　定时器

定时器是用于产生时间序列的，这些时间序列可用于等待、监控、测量时间间隔或者产生脉冲。定时器的数目由 CPU 决定。不同的 CPU 支持 32～512 个定时器。定时函数存放在 CPU 的系统存储器中。

定时器的种类有：脉冲定时器（SP）、扩展脉冲定时器（SE）、接通延时定时器（SD）、保持型接通延时定时器（SS）和断开延时定时器（SF）。五种类型定时器的时序如图 4-31 所示，图中 $t$ 为预设的时间值。

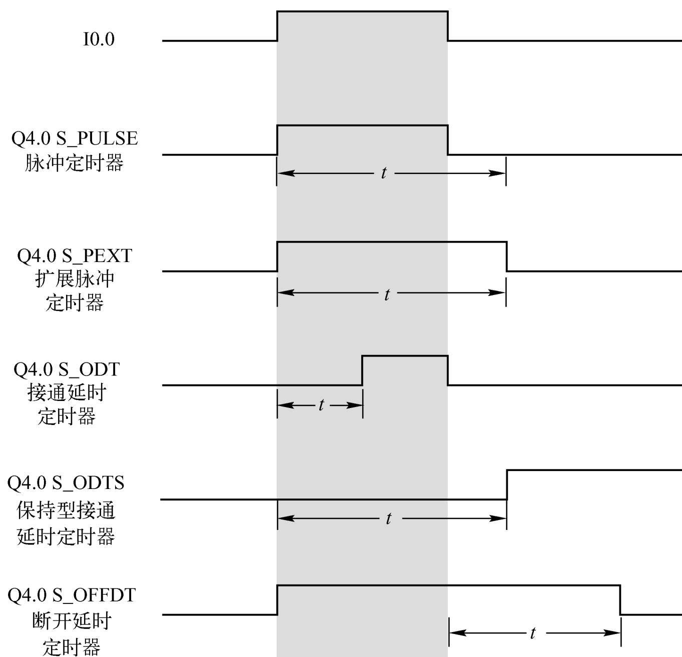



图 4-31　五种类型定时器的时序

**1. 定时器的精确性**

在使用定时器之前，要了解定时的精确性。因为当创建了一个持续几秒的定时器时，并没有特别考虑定时器的准确性，而当要创建一个毫秒级定时器，我们就必须关心一下这个问题。

一个定时器存在两种误差：输入误差、输出误差。总误差为两者之和。

① 输入误差：当定时器在扫描周期中开始工作，且当 PLC 正好扫描完输入状态之后，此时定时器允许工作的输入状态开通，这时 PLC 只能在下一个循环扫描时才能捕捉到这一开通值。此时输入误差最大。

② 输出误差：输出误差也同样。当 PLC 读到定时器时间到的信号后，必须执行完其余程序后，才能真正刷新输出，此时输出误差产生。

由于上述问题，定时器可能产生的最大误差为：2 个扫描周期 +1 个程序执行时间。

这意味着即使我们有定时器（毫秒级），但是并不能保证在几毫秒内准确。比如，当扫描周期是 5 ms 时，最好不要使用小于 5 ms 的定时器。尤其在高速、高精度场合，这种误差是非常明显和致命的。解决方法是使用外部的硬件定时器。

**2. 定时器的使用**

定时器有其存储区域，每个定时器有一个 16 位的字和一个二进制的值。其中定时器的字用于存放当前定时值。二进制的值表示定时器接点的状态。

如何使用定时器呢？首先必须了解两个问题：

① 如何起动和停止定时器。PLC 中的定时器相当于时间继电器。在使用时间继电器时，当时间继电器被起动，若定时时间到，则继电器的接点动作，同样，当时间继电器的线圈断电时，接点也动作。在 S7 中定时器与时间继电器的工作原理类似，但功能更加丰富。可以完成以下几种功能：设定定时时间、起动定时器、复位定时器、查看定时的剩余时间。

至于起动和停止定时器，在梯形图中，定时器的 S 端可以使能定时器。定时器的 R 端可以复位定时器。

② 如何设定定时时间。S7 中定时时间由时基和定时值组成，定时时间为时基和定时值的乘积。在定时器开始工作后，定时值不断递减，递减至零表示时间到，定时器会相应动作。

如图 4-32 所示为定时器字的格式，其中第 12 ~ 13 位是定时器的时基。所谓的时基是时间基准的简称。定时时间值是以三位 BCD 码格式存放，位于定时器字的第 0 ~ 11 位。使用范围是 0 ~ 999。表 4-7 给出时基与相应的定时范围。

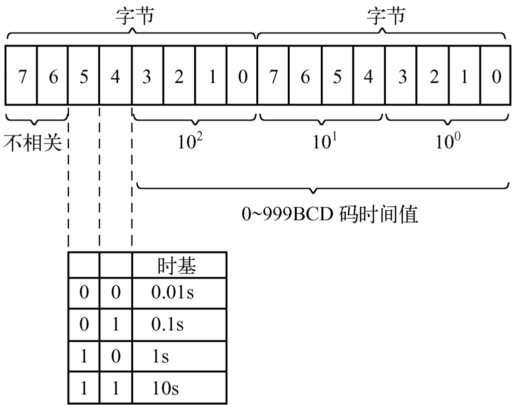



图 4-32　定时器字的格式

定时时间有两种表达方式：

① 十六进制数。格式为：W#16#wxyz，其中 w 是时间基准，xyz 是 BCD 码格式的时间值。这里，时基越小，分辨率越高；时基越大，则分辨率越低，但定时时间越长。

② S5 时间格式。格式为：S5T#aH_bM_cS_dMS，其中，a 表示小时，b 表示分钟，c 表示秒，d 表示毫秒。例如：S5T#1H_13M_8S 表示时间为 1 h13 min8 s。这里时基是由 CPU 自行选定的，原则是在满足定时范围的要求下选择最小时基。

表 4-7　时基与相应的定时范围

| 时基 | 时基的二进制代码 | 分辨率/s | 定时范围 |
|---|---|---|---|
| 10 ms | 0 0 | 0. 01 | 10 MS ~ 9S_990 MS |
| 100 ms | 0 1 | 0. 1 | 100 MS ~ 1 M_39 S _900 MS |
| 1 s | 10 | 1 | 1 S ~ 16 M_39 S |
| 10 s | 11 | 10 | 10 S ~ 2H_46 M_30 s |

### 3. 方块指令的参数含义及语句表指令含义

S5 定时器的梯形图和功能块图方块指令的参数表如表 4-8 所示。

表 4-8　S5 定时器方块指令的参数表

| 参数 | 数据类型 | 存储区 | 说明 |
|---|---|---|---|
| No. | TIMER | T | 定时器标识号 |
| S | BOOL | I、Q、M、D、L | 起动输入端 |
| TV | S5TIME | I、Q、M、D、L | 预置时间值 |
| R | BOOL | I、Q、M、D、L | 复位输入端 |
| Q | BOOL | I、Q、M、D、L | 定时器状态 |
| BI | WORD | I、Q、M、D、L | 当前时间值（整数格式） |
| BCD | WORD | I、Q、M、D、L | 当前时间值（BCD 格式） |

定时器的语句表指令含义如表 4-9 所示。

表 4-9　定时器的语句表指令

| 指令 | 说明 |
|---|---|
| FR | 允许定时器再起动 |
| L | 将定时器的时间值（整数）装入累加器 1 中 |
| LC | 将定时器的时间值（BCD）装入累加器 1 中 |
| R | 复位定时器 |
| SD | 接通延时定时器 |
| SE | 扩展脉冲定时器 |
| SF | 断开延时定时器 |
| SP | 脉冲定时器 |
| SS | 保持型接通延时定时器 |

### 4. 脉冲定时器

脉冲定时器的功能块图和梯形图如图 4-33 所示。

脉冲定时器时序波形如图 4-34 所示，其中 $t$ 为设定的定时时间。

从脉冲定时器时序波形可以看出：当 S 端从“0”变到“1”（RLO 的正跳沿），脉冲定时器起动。在定时时间到达之前，如果输入 S 端从“1”变到“0”，则定时器结束定时。同样在 R 端从“0”变到“1”（RLO 的正跳沿），无论定时时间是否到，定时器复位为零。之所以叫脉冲定时器，是因为在脉冲宽度期间，定时器能正常定时，否则，就复位至初始状态。

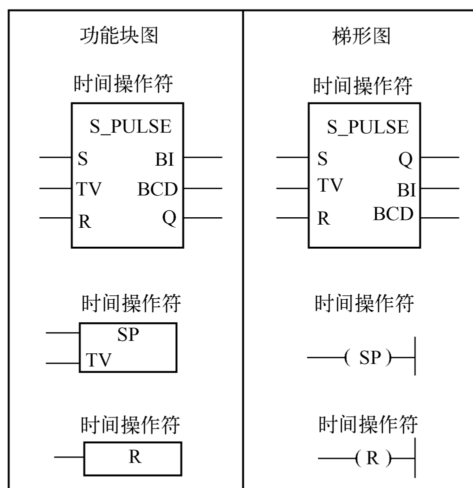


图 4-33　脉冲定时器的功能块图和梯形图

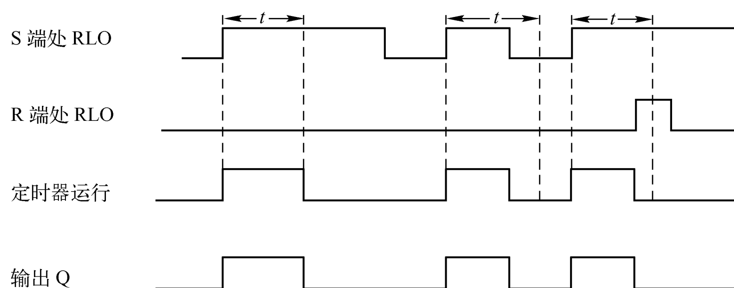


图 4-34　脉冲定时器时序波形

脉冲定时器方块指令在梯形图和功能块图中的具体应用如图 4-35a、b 所示。

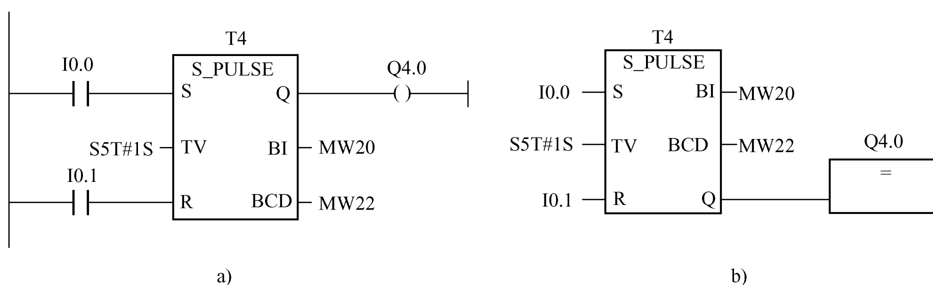


图 4-35　脉冲定时器方块指令在梯形图和功能块图中的具体应用

说明：当 I0.0 接通时，其 RLO 的正跳沿起动脉冲定时器 T4，定时时间为 1 s，I0.1 复位定时器。当前定时值通过 MW20 和 MW22 输出。当定时器工作时（RLO 在定时期间保持为 1），输出 Q4.0 接通。

实现与上述功能相同的语句表程序为：

```
A   I0.0        //起动定时器 T4
L   S5T#1S      //装载定时时间
SP  T4          //脉冲定时器
A   I0.1        //复位定时器 T4
R   T4
L   T4          //装载当前时间值(整数格式)
T   MW20
LC  T4          //装载当前时间值(BCD 码格式)
T   MW22
A   T4
=   Q4.0        //检测定时器状态
```

说明：SP 指令表示脉冲定时器，通过装载指令 L 设定 S5T 格式的定时时间值，装载指令 L 将不带时基的十六进制整数格式的定时器当前值 MW20 传送到累加器 1 的低字中；而 LC 则是将 BCD 码格式的定时器当前值 MW22 和时基装入累加器 1 的低字中。定时器的复位指令为 I0.1，起动指令为 I0.0，输出为 Q4.0。

脉冲定时器线圈指令在梯形图和功能块图中的具体应用如图 4-36a、b 所示。可见线圈指令相对简单，在定时器只需要基本功能时，使用线圈指令更为合适。

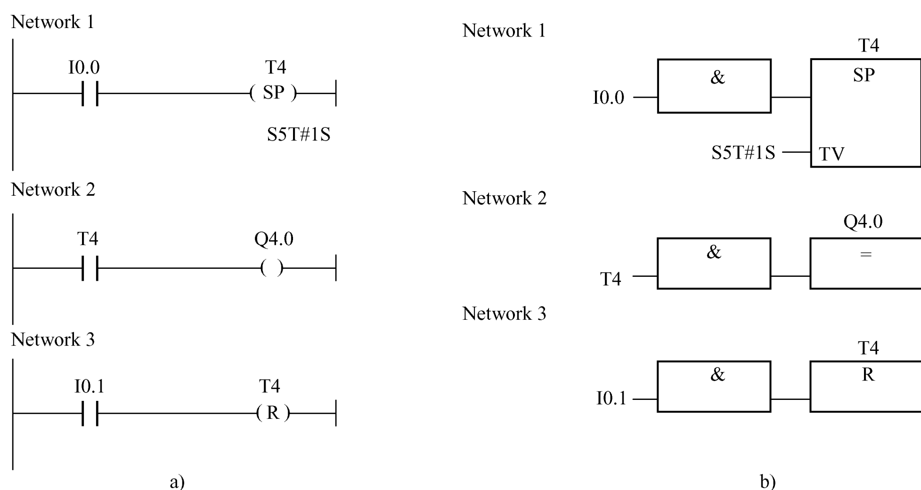



图 4-36　脉冲定时器线圈指令在梯形图和功能块图中的具体应用

说明：I0.0 的正跳沿起动脉冲定时器 T4（定时时间为 1 s），T4 定时期间（RLO 在定时期间保持为 1）使输出 Q4.0 接通，I0.1 复位定时器。

**例 6**　用脉冲定时器设计一个周期振荡电路，振荡周期为 5 s，占空比为 2∶5。

如图 4-37 所示为用脉冲定时器设计的振荡电路的梯形图程序。

说明：在设计中，我们用 T1 和 T2 分别定时 2 s 和 3 s，用 I0.0 起动振荡电路。由于是周期振荡电路，所以 T1 和 T2 必须互相起动。在程序的 Network1 中，T2 需用常闭节点，否则，T1 无法起动。在 Network2 中，T1 工作期间，T2 不能起动工作。所以 T1 需用常闭节点来起

动 T2。即当 T1 定时时间到时，T1 的常闭节点接通，从而产生 RLO 上跳沿，起动 T2 定时器。如此循环，在 Q4. 0 端形成振荡电路。

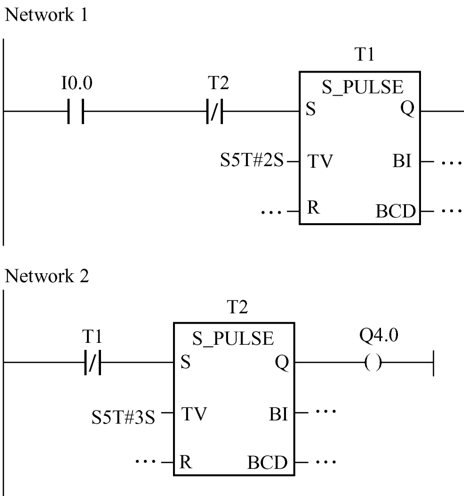



图 4-37　用脉冲定时器设计的振荡电路的梯形图程序

**5. 扩展脉冲定时器**

扩展脉冲定时器的时序波形如图 4-38 所示，其中 $t$ 为设定的定时时间值。

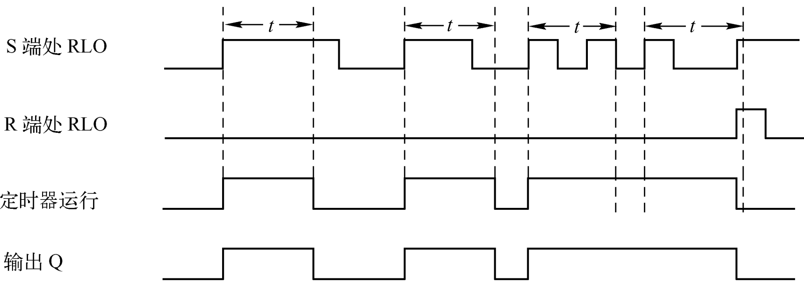



图 4-38　扩展脉冲定时器的时序波形图

从扩展脉冲定时器的时序波形可以看出：当输入 S 端从“0”变到“1”（RLO 的正跳沿），扩展脉冲定时器起动。定时器将持续工作而不受起动输入端 S 端负跳沿的影响。在定时时间到达之前，如果输入 S 端从“0”变到“1”，则扩展脉冲定时器再次起动。输出端 Q 在定时器工作期间始终为“1”。

我们将脉冲定时器和扩展脉冲定时器相比较可以发现：扩展脉冲定时器即使在脉冲宽度不够定时宽度时，也能使定时器运行至定时时间结束。同时从时序波形可以发现：在起动端不断由“0”变为“1”时，只要定时时间未到，则定时器反复起动，输出 Q 在此期间始终为“1”。

S5 扩展脉冲定时器方块指令在梯形图和功能块图中的具体应用如图 4-39a、b 所示。

说明：当 I2. 1 接通时，其 RLO 的正跳沿起动扩展脉冲定时器 T1，T1 定时时间为 10 s，I2. 2 复位定时器。当前定时值通过 MW10 和 MW12 输出。当定时器 T1 定时工作时，输出

Q4.0 接通。

实现与上述功能相同的语句表程序为：

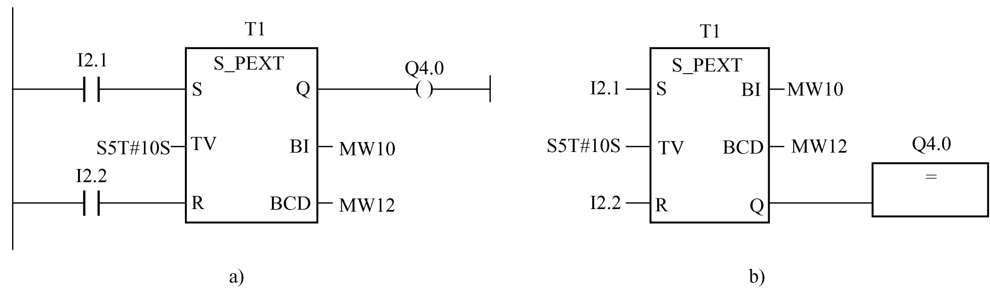



图 4-39　扩展脉冲定时器方块指令在梯形图和功能块图中的具体应用

```
A     I2. 1          //起动定时器 T1
L     S5T#10S        //设置时间 10 s
SE    T1             //扩展脉冲定时器
A     I2. 2
R     T1             //复位定时器 T1
L     T1             //装载当前时间值(整数格式)
T     MW10
LC    T1             //装载当前时间值(BCD 码格式)
T     MW12
A     T1             //检查定时器状态
=     Q4. 0
```

说明：SE 指令表示扩展脉冲定时器。其余语句表含义同脉冲定时器。

扩展脉冲定时器线圈指令在梯形图和功能块图中的具体应用如图 4-40a、b 所示。

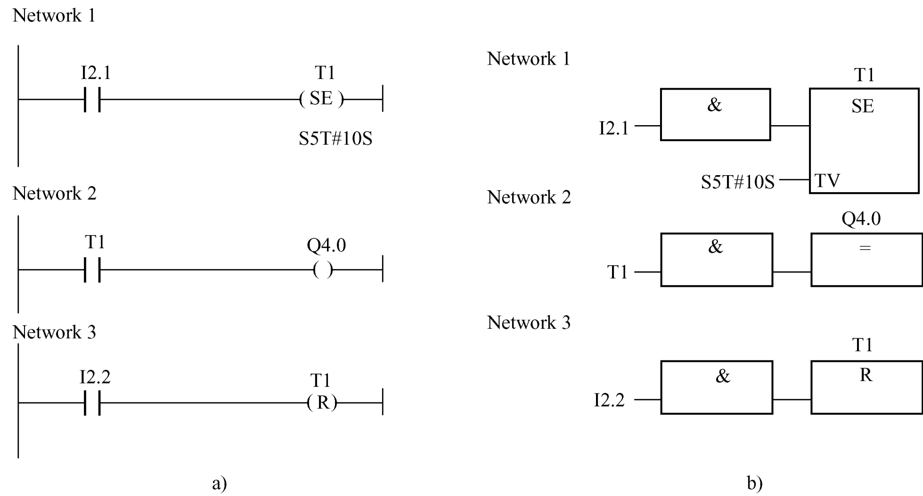



图 4-40　扩展脉冲定时器线圈指令在梯形图和功能块图中的具体应用

说明：当 I2.1 接通时，其 RLO 的正跳沿起动扩展脉冲定时器 T1，T1 定时时间为 10 s，I2.2 复位定时器。当定时器 T1 定时工作时，输出 Q4.0 接通。

**例 7** 设计频率监视器，其特点是频率低于下限，则指示灯 Q4.0 亮，"确认"按钮 I0.1 使指示灯复位。监控频率为 0.5 Hz，由 M10.0 提供（M10.0 的具体设置方法参见 8.2 节）。

频率监控电路的梯形图程序如图 4-41 所示。

说明：在设计中，由于扩展脉冲定时器的特点：时间未到时，若输入 S 端反复正跳变，则定时器反复起动，输出始终为 1，直至定时时间到为止，在此使用非常合适。若监控频率为 0.5 Hz，则使用定时时间为 2 s 的定时器。在频率正常的情况下，0.5 Hz 的频率反复起动 2 s 的定时器，使输出始终为高电平。当频率变低，脉冲时间间隔变大时，2 s 的定时器可以计时完毕，此时输出变为低电平。监控指示灯 Q4.0 亮。

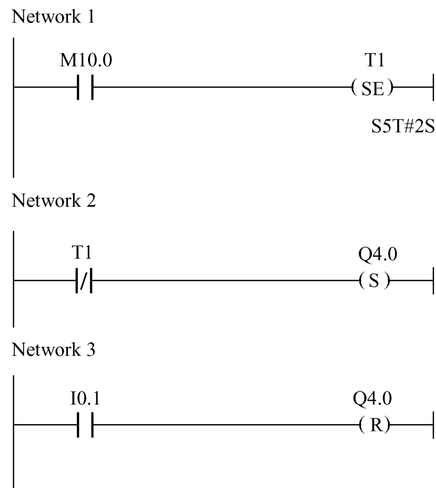



图 4-41 频率监控电路的梯形图程序

程序的语句表实现为：

```
A     M10.0
L     S5T#2S
SE    T1
AN    T1
S     Q4.0
A     I0.1
R     Q4.0
```

**6. 接通延时定时器**

接通延时定时器的时序波形如图 4-42 所示。

接通延时定时器的特点简而言之就是定时时间到，输出才接通。当输入 S 端从"0"变到"1"（RLO 的正跳沿），接通延时定时器起动。如果在定时期间，S 端状态保持不变，则定时时间到后，使输出 Q 变为"1"。如果输入 S 端从"1"变到"0"，则接通延时定时器输出 Q 变为"0"。当输入 R 端从"0"变到"1"，则不论定时器工作否，定时器均复位。

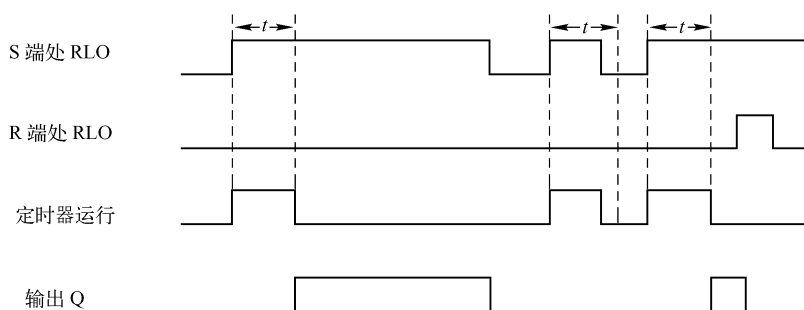



图 4-42　接通延时定时器的时序波形

S5 接通延时定时器方块指令在梯形图和功能块图中的具体应用如图 4-43a、b 所示。

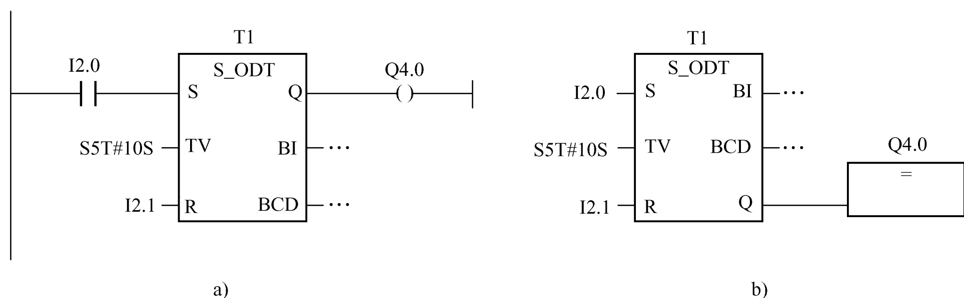



图 4-43　S5 接通延时定时器方块指令在梯形图和功能块图中的具体应用

说明：当 I2.0 接通时，其 RLO 的正跳沿起动接通延时定时器 T1，T1 定时时间为 10 s，I2.1 复位定时器。当定时器 T1 定时结束时（RLO 在定时期间保持为 1），输出 Q4.0 接通。

实现与上述功能相同的语句表程序为：

```
A    I2.0
L    S5T#10S        //设置定时时间
SD   T1             //接通延时定时器
A    I2.1
R    T1             //复位定时器 T1
A    T1             //检测 T1 状态
=    Q4.0
```

说明：SD 指令为接通延时定时器。

接通延时定时器线圈指令在梯形图和功能块图中的具体应用如图 4-44a、b 所示。

说明：当 I2.0 接通时，其 RLO 的正跳沿起动接通延时定时器 T1，T1 定时时间为 10 s，I2.1 复位定时器。当定时器 T1 定时工作时（RLO 在定时期间保持为 1），输出 Q4.0 接通。

**例 8**　用接通延时定时器设计一个周期振荡电路，振荡周期为 5 s，占空比为 2：5。

前面我们用脉冲定时器设计了周期振荡电路，现在用接通延时定时器来再次实现之。用接通延时定时器线圈指令设计的振荡电路程序如图 4-45 所示。

说明：与脉冲定时器的设计电路相比，在程序的 Network2 中，T1 是常开接点。在接通延时定时器定时时间到时，T1 工作结束，输出高电平，其上跳沿起动定时器 T2，这样 T1

和 T2 就可以互相起振。而脉冲定时器的 T1 是常闭接点，在 T1 不工作期间，输出为低电平，常闭接点接通，此时，T2 开始定时。

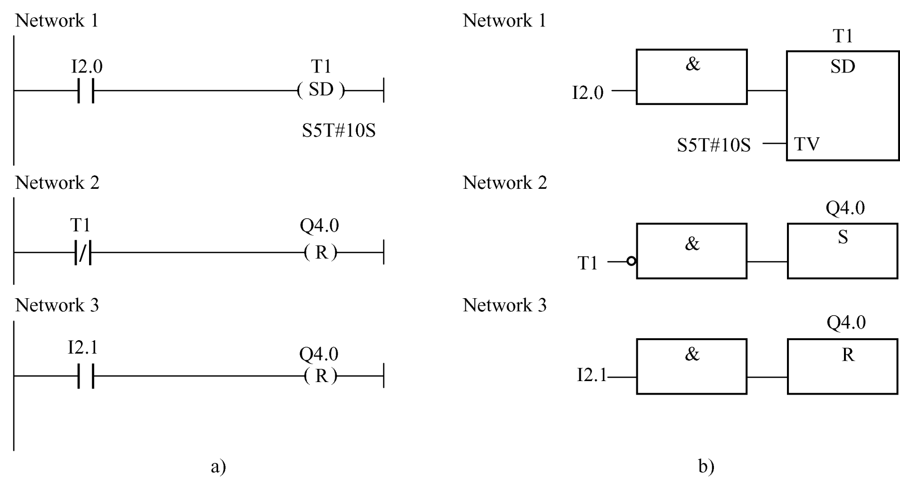



图 4-44　接通延时定时器线圈指令在梯形图和功能块图中的具体应用

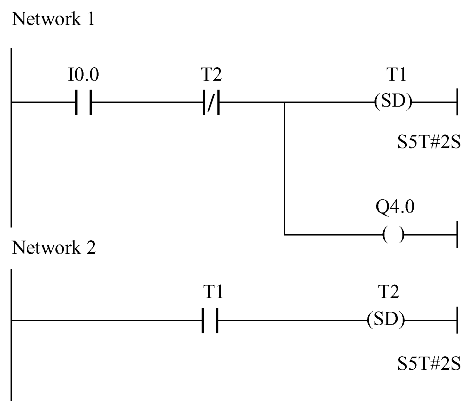



图 4-45　用接通延时定时器线圈指令设计的振荡电路程序

STL 语句实现如下：

```
A    I0. 0
AN   T2
L    S5T#2S
SD   T1
=    Q4. 0
A    I0. 0
A    T1
L    S5T#3S
SD   T2
```

**7. 保持型接通延时定时器**

保持型接通延时定时器的时序波形如图 4-46 所示。我们将保持型接通延时定时器和接通延时定时器相比较可以发现：保持型接通延时定时器即使在脉冲宽度不够定时宽度时，也能使定时器运行至定时时间结束。同时从时序波形可以发现：在起动端不断由“0”变为“1”时，只要定时时间未到，则定时器反复起动，输出 Q 在此期间始终为“0”。直至定时时间到，输出才变为“1”。

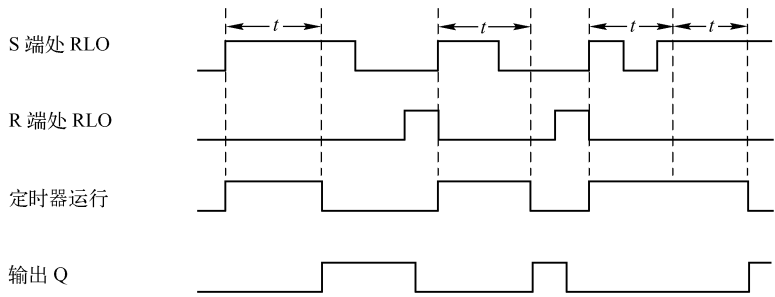



图 4-46　保持型接通延时定时器的时序波形

S5 保持型接通延时定时器方块指令在梯形图和功能块图中的具体应用如图 4-47a、b 所示。

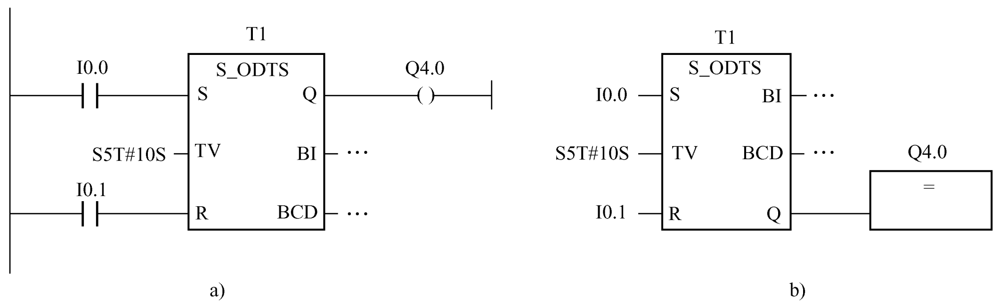



图 4-47　S5 保持型接通延时定时器方块指令在梯形图和功能块图中的具体应用

说明：当 I0.0 接通时，其 RLO 的正跳沿起动保持型接通延时定时器 T1，T1 定时时间为 10 s，I0.1 复位定时器。当定时器 T1 定时结束时，输出 Q4.0 接通。

实现与上述功能相同的语句表程序为：

```
A    I 2.0
FR   T1          //起动定时器 T1
A    I 0.0
L    S5T#10 s    //设置定时时间
SS   T1          //保持型接通延时定时器
A    I 0.1
R    T1          //复位定时器
A    T1          //检测 T1 状态
```

=　　Q 4.0

说明：SS 表示保持型接通定时器。

保持型接通延时定时器线圈指令在梯形图和功能块图中的具体应用如图 4-48a、b 所示。

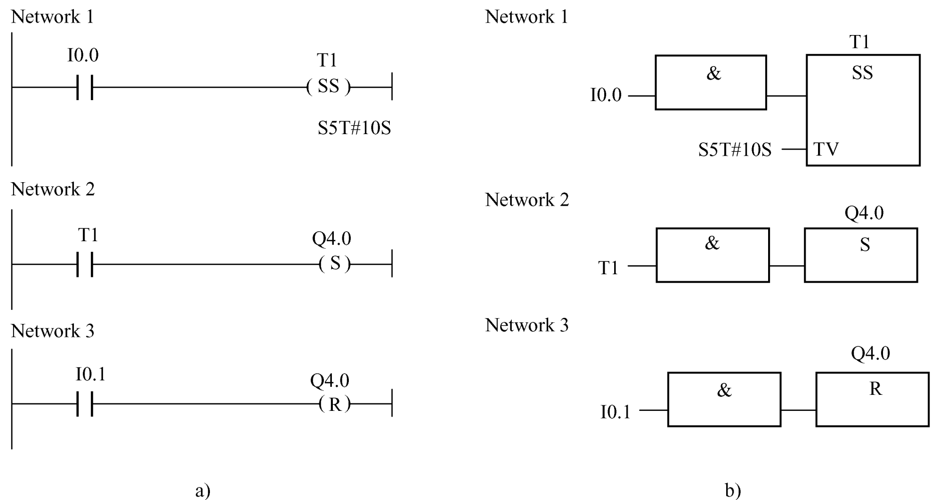



图 4-48　保持型接通延时定时器线圈指令在梯形图和功能块图中的具体应用

说明：当 I0.0 接通时，其 RLO 的正跳沿起动接通延时定时器 T1，T1 定时时间为 10 s，I0.1 复位定时器。当定时器 T1 定时结束时，输出 Q4.0 接通。

**8. 断开延时定时器**

断开延时定时器的时序波形如图 4-49 所示。

断开延时定时器的特点是在 S 端上跳沿时，输出为“1”，在 S 端下跳沿，定时器起动，直至定时时间到，输出在此期间始终为“1”。即所谓 S 端断开，才开始延时定时。

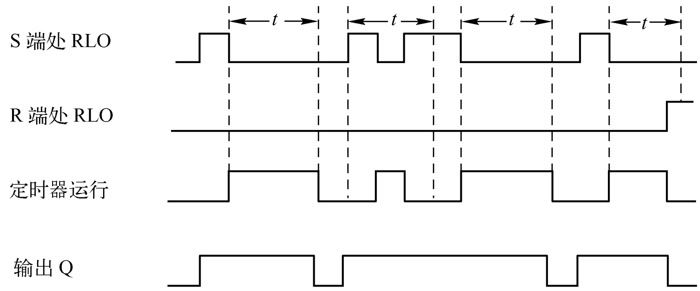



图 4-49　断开延时定时器的时序波形

S5 断开延时定时器方块指令在梯形图和功能块图中的具体应用如图 4-50a、b 所示。

说明：当 I0.0 接通时，其 RLO 的正跳沿起动断开延时定时器 T1，T1 定时时间为 10 s，I0.1 复位定时器。输出 Q4.0 从 RLO 的正跳沿起到定时结束止处于接通状态。

实现与上述功能相同的语句表程序为：

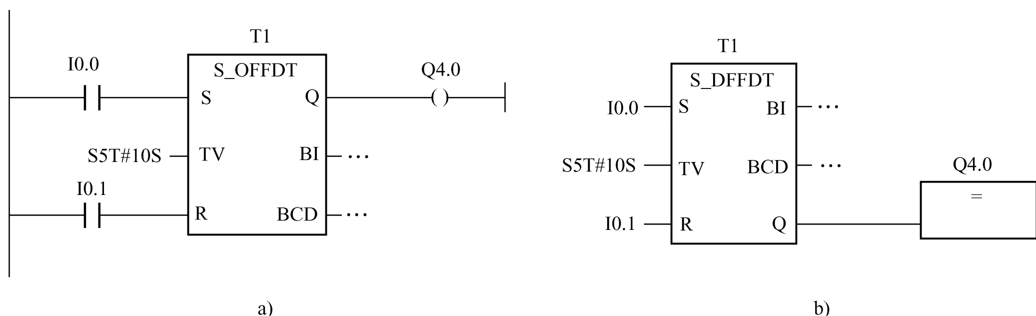



图 4-50　S5 断开延时定时器方块指令在梯形图和功能块图中的具体应用

```
A     I 2.0
FR    T1          //起动定时器 T1
A     I 0.0
L     S5T#10 s    //设置定时时间
SF    T1          //断开延时定时器
A     I 0.1
R     T1          //复位定时器
A     T1          //检测 T1 状态
=     Q 4.0
```

说明：SF 指令表示断开延时定时器。

断开延时定时器线圈指令在梯形图和功能块图中的具体应用如图 4-51a、b 所示。

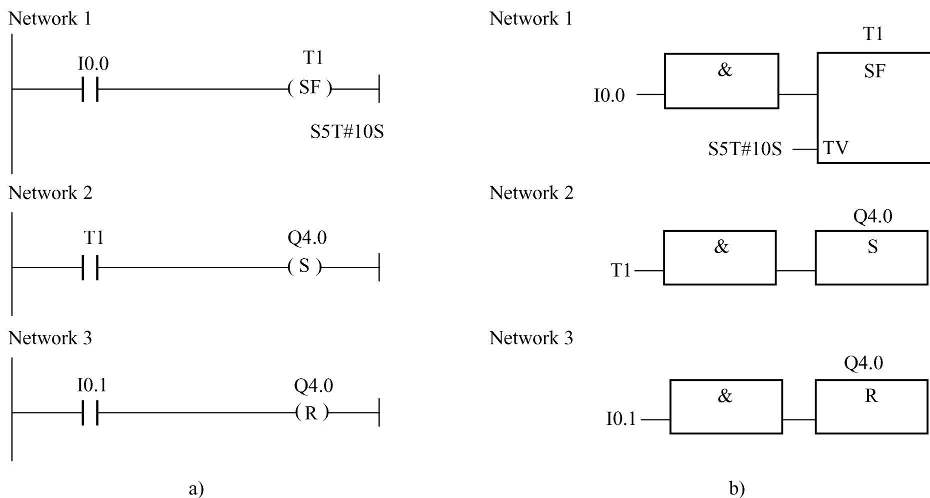



图 4-51　断开延时定时器线圈指令在梯形图和功能块图中的具体应用

说明：当 I0.0 接通时，其 RLO 的正跳沿起动断开延时定时器 T1，T1 定时时间为 10 s，I0.1 复位定时器。输出 Q4.0 从 RLO 的正跳沿起到定时结束止处于接通状态。

定时器使用小结：

① S7 -300 的定时器种类比较多，编程时恰当的使用可以简化程序。

② 当定时时间比较长，可以用几个定时器完成定时时间，采取依次起动下一个定时器的做法，达到定时时间的叠加，或者采取和计数器联合的方法实现。

③ 定时振荡在实际应用中也比较常见，需要掌握；振荡也可以用系统内部时钟来实现。在第 8 章会有所提及。

④ 对于定时时间很短或者精度要求比较高时，需要外部时钟实现。

### 4.5.2 计数器

计数器的任务是完成计数功能，可以实现加法计数和减法计数，计数范围是 0 ~ 999。它的种类以此类推有三种：加法计数器、减法计数器和加减可逆计数器。

这里有个问题比较令人费解，既然 CPU 中有计数器，为什么又有高速计数器模块呢？因为高速计数器是“硬件”装置，而上述的计数器是“软件”装置，“硬件”装置的计数器独立于 CPU 的扫描周期，具有较高的精度。比如：CPU 的扫描时间是 2 ms，脉冲计数为每 4 ms 计数一次。如果用户需要 3 ms 计数一次，为保证精度，必须选用高速计数器。

**1. 计数器的使用**

与定时器相同，计数器在 CPU 中也占有一个存储区，称为计数器计数值存储区。每个计数器占用 2 个字节，称为计数器字。计数器字格式如图 4-52 所示，其中 BCD 码格式计数值在计数器字的第 0 ~ 11 位，范围是 0 ~ 999，用 C#来表示；而二进制格式的计数值在计数器字的第 0 ~ 9 位。

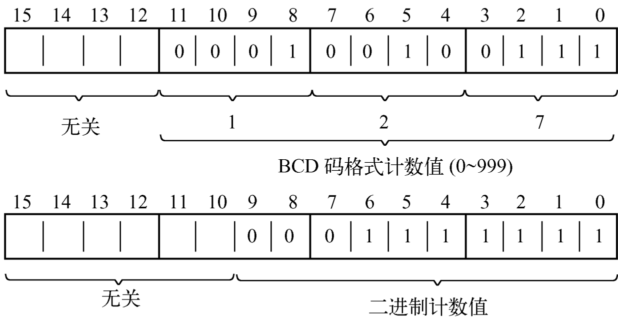



图 4-52　计数器字格式

当 BCD 码格式计数值达到上限 999 时，累加停止；反之，当计数值达到下限时，计数值保持为零。当对计数器进行初始设定时，累加器 1 低字中的内容装载入计数器字中。

在使用计数器之前，用户必须注意几个问题：

1）计数脉冲从何而来，这涉及计数器的起动问题。

2）在开始动作之前，需要计多少个数。这涉及赋值问题。比如，我们要将 5 个货物包装在一个箱子中。那就需要赋值为 5，并使用减法计数器。

3）如何复位计数器，让它重新开始计数。比如 5 个货物装好后，需要再装另外 5 个货物，此时必须重新起动计数器。

4）如何实现现场监控当前计数值。

**2. 计数器指令及用法**

如表 4-10 所示是三种计数器的梯形图参数的含义。

**表 4-10　计数器梯形图参数表**

（a）

| 可逆计数器 | 加计数器 | 减计数器 |
|---|---|---|
| C no.<br>S_CUD<br>CU　Q<br>CD<br>S　CV<br>PV　CV_BCD<br>R | C no.<br>S_CU<br>CU　Q<br>S<br>PV　CV<br>CV_BCD<br>R | C no.<br>S_CD<br>CD　Q<br>S<br>PV　CV<br>CV_BCD<br>R |

（b）

| 参数 | 数据类型 | 存储区 | 说明 |
|---|---|---|---|
| no. | COUNT | C | 计数器标识符 |
| CU | BOOL | I、Q、M、D、L | 加计数器输入 |
| CD | BOOL | I、Q、M、D、L | 减计数器输入 |
| S | BOOL | I、Q、M、D、L | 计数器初值预置 |
| PV | WORD | I、Q、M、D、L | 初始值 BCD 码 |
| R | BOOL | I、Q、M、D、L | 复位输入端 |
| Q | BOOL | I、Q、M、D、L | 计数器状态输出 |
| CV | WORD | I、Q、M、D、L | 当前计数值（整数） |
| CV_BCD | WORD | I、Q、M、D、L | 当前计数值（BCD） |

加减可逆计数器梯形图和功能块图的框图的用法如图 4-53a、b 所示。

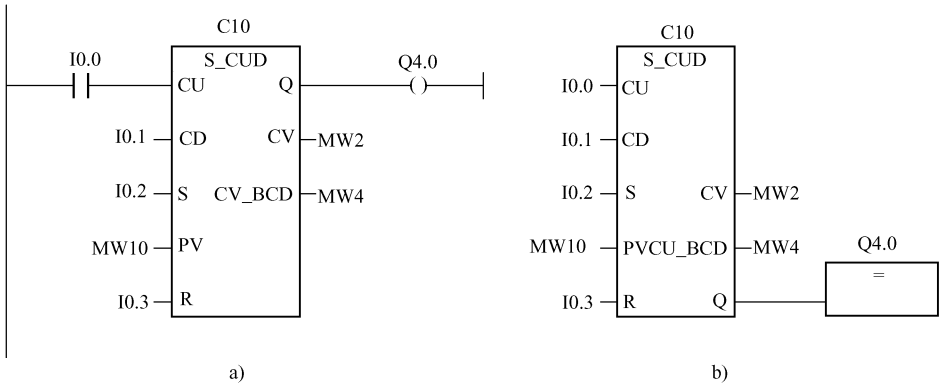



图 4-53　加减可逆计数器梯形图和功能块图的框图的用法

说明：当输入 I0.2 接通时，将计数值 MW10 赋给计数器。在输入信号 I0.0（CU：加信号端）的每一个正跳沿，计数器 C10 的值加 1，直至 999；而在输入信号 I0.1（CD：减信号端）的每一个正跳沿，计数器 C10 的值减 1，直至 0。只要计数值不为零，则输出 Q 值为“1”。当前计数值通过 MW2 以整数格式显示，通过 MW4 以 BCD 码格式显示。

实现如上相同功能的计数器的语句表指令为：

```
A     I0.0          //在 I0.0 的上升沿
```

```
CU    C10          //计数器 C10 值加 1
A     I0.1         //在 I0.1 的上升沿
CD    C10          //计数器 C10 值减 1
A     I0.2         //在 I0.2 的上升沿
L     MW10         //预置值装入累加器 1 中
S     C10          //将预置值装入计数器中
A     I0.3         //在 I0.3 为 1 时
R     C10          //复位 C10
L     C10          //将 C10 的二进制计数当前值装入累加器 1
T     MW2          //将累加器 1 的内容传入 MW2 中
LC    C10          //将 C10 的 BCD 计数当前值装入累加器 1
T     MW4          //将累加器 1 的内容传入 MW4 中
A     C10          //如果 C10 值不为零
 =Q4.0/设置 Q4.0 为 1
```

说明：CU 表示加计数器指令，CD 表示减计数器指令。

通过上述计数器的用法说明，前面所提到的几点问题现在可以解决了。

加减计数器的梯形图及功能块图的线圈指令及用法如图 4-54a、b 所示。

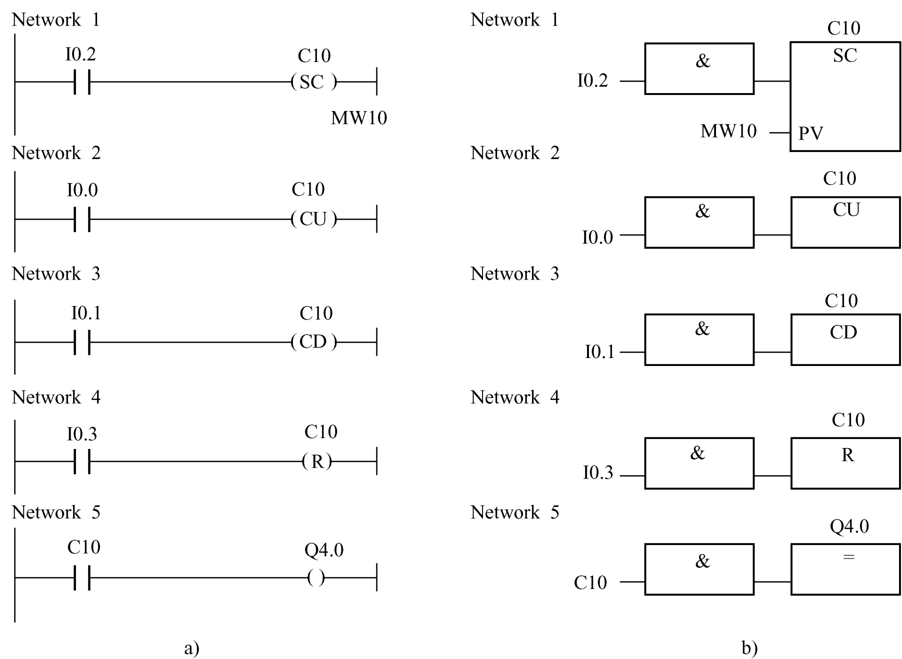



图 4-54　加减计数器的梯形图及功能块图的线圈指令及用法

说明：当输入 I0.2 为 1 时，将计数值 MW10 赋给计数器。在输入信号 I0.0 的每一个正跳沿，计数器 C10 的值加 1；而在输入信号 I0.1 的每一个正跳沿，计数器 C10 的值减 1，直至为 0。只要计数值不为零，则输出 Q4.0 值为“1”。

**例 9**　用计数器扩展定时器的定时范围。要求：I0.0 为启停开关，定时范围为 12 h。12 h之后，将电磁阀 Q4.0 打开。

分析：我们知道定时器最长的时间是 9990 s，约 2 个多小时。为了实现 12 h 的定时功能，我们先设计一周期振荡电路，其中接通延时定时器 T1 和 T2 的定时时间均为 7200 s，这样振荡周期为 4 h，如果结合一个初始值为 3 的减法计数器，每隔 4 h 触发，则在减计数器计数值减至零时，相当于经过了 12 h。

定时器和计数器结合扩展定时范围的功能块图程序如图 4-55 所示。

说明：程序的 Network1 和 Network2 构成振荡电路，周期为 4 h。在 Network3 中，赋值语句前加正边沿触发指令，保证计数器初值只赋值一次。而在 Network4 中，用 T1 的负边沿触发指令来起动计数器，因为在开始 2 h，T1 的输出为“0”，2 h 后，延时时间到，T1 的输出为“1”，这样经过 4 h，T1 才能出现负跳沿。如果用户用 T1 的正边沿指令将少 2 个 h。所以，在编程中一定要注意细节。当减计数器计数值为 0 时，定时时间已经到 12 h。在 Network6 中，以 C0 的常闭接点和 I0.0 起动按钮的并联控制输出 Q4.0。

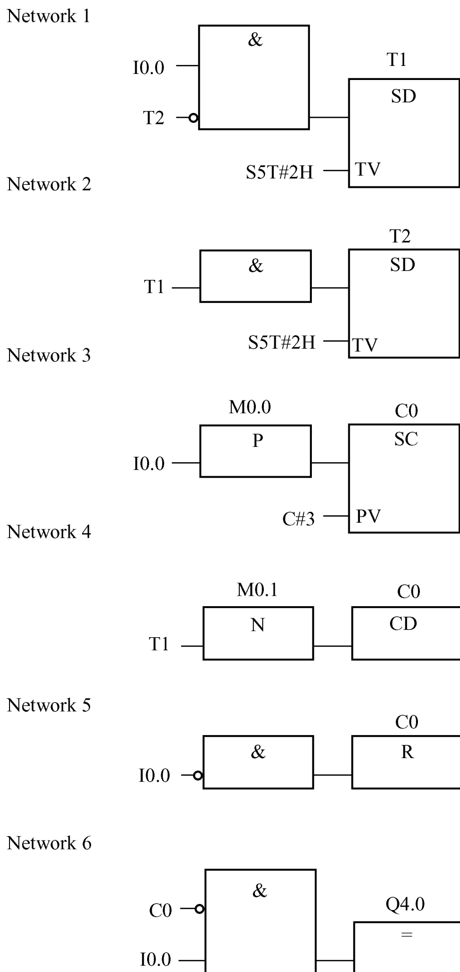



图 4-55　定时器和计数器结合扩展定时范围的功能块图程序

## 习题Ⅱ

1. 设计三台电动机顺序起动和停止电路。要求在手动状态下，按手动起动按钮 I0.0，第一台电动机 Q4.0 起动运行，5 s 后第二台电动机 Q4.1 开始运行，6 s 后第三台电动机 Q4.2 开始运行。按手动停止按钮 I0.1，则第三台电动机先停，3 s 后第二台停，再过 3 s 后第一台电动机停。

2. 电动机顺序起动和停止的电路设计中，在自动运行按钮 I0.3 工作状态下，每间隔 5 s 一、二、三台电动机顺序起动，运行 20 s 后，每间隔 5 s 一、二、三台电动机顺序停止，如此循环。在紧急事故状态下（I0.4），三台电动机同时停止。

3. 试分析并完成输出 Q4.0 的时序图，了解计数器的使用方法。输入波形如图 4-56 所示。STL 程序如下：

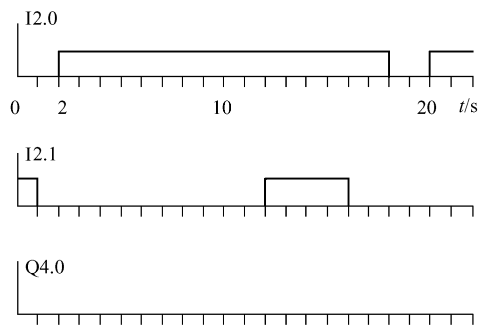



图 4-56　习题 3 图

```
AN    I2.0
A     I2.1
L     C#3
S     C1
A     I2.0
A     I2.1
CU    C1
A     I2.0
CD    C1
AN    I2.0
AN    I2.1
R     C1
L     C1
T     MW0
A     C1
=     Q4.0
```

4. 试设计一个照明灯的控制电路。当按下 I0.0 按钮，照明灯 Q4.0 可发光 30 s，如果在这段时间内又有人按下按钮，则时间间隔从头开始。这样可确保在最后一次按完按钮后，灯光可维持 30 s 照明。

5. 要求在 3 个不同地方控制某台电动机的起动和停止。每个地方都有起动和停止按钮。在任意一个地方按下起动按钮，则电动机起动旋转，按钮弹起，电动机保持旋转；在任意一个地方按下停止按钮，则电动机停止旋转。根据要求，建立 I/O 变量表，并编制控制梯形图程序。

6. 用 I0.0 控制 Q0.0、Q0.1 和 Q0.2，要求：若 I0.0 闭合三次，Q0.0 亮，I0.0 再闭合三次，Q0.1 亮，若再闭合三次，Q0.2 亮，之后，I0.0 闭合一次，Q0.0、Q0.1 和 Q0.2 都灭，如此循环进行。

## 4.6　数据处理功能指令

数据处理功能指令主要包括装载和传输指令、比较指令、转换指令及移位和循环指令等。

### 4.6.1　装载和传输指令

装载指令 L（LOAD）和传输指令 T（TRANSFER）用于在存储区之间或存储区与过程输入、输出之间交换数据。装载和传输需要累加器的配合，累加器在 CPU 中是存储数据的临时寄存器，就像临时中转站。在 S7－300 中，L 指令执行时，将源操作数装载入累加器 1 中，而累加器原有的数据移至累加器 2 中，累加器 2 中的数据被覆盖；T 指令将累加器 1 中的数据内容写入目的存储器中。S7－300 中，有 2 个累加器，而 S7－400 中累加器有 4 个，在每次执行装载指令时，只有第 4 个累加器的内容被覆盖。

装载和传输指令的意义何在呢？换句话说："我们为什么要获得数据呢?"，答案很简单：比如在使用 A/D 转换模块时，A/D 模块需要将模拟信号（电流、电压等信号）变成 PLC 能够理解的信号（数字信号"0"或"1"）。制造厂商很自然地将这些数据存储起来。我们必须在下一个采样周期到来之前将数据移走。否则，数据将被覆盖。同样我们可能还需要存储某些常数值、获得二进制值或者进行简单的数学运算，这些都需要装载和传输指令。

L 和 T 指令可以对字节、字和双字数据进行操作。装载和传输指令操作有三种寻址方式：立即寻址、直接寻址和间接寻址。这些寻址方式在前面已经有所介绍。

**1. 几种典型的装载和传输指令**

（1）对累加器 1 的装载和传输指令

```
L   +8          //将立即数装载入累加器 1 中
L   IB[DID 8]   //将数据双字 DID8 所指的输入字节装载入累加器 1 中
T   QB10        //将累加器 1 的内容传输给输出 QB10
T   MW14        //将累加器 1 的内容传输给存储字 MW14
T   DBD2        //将累加器 1 的内容传输给数据双字 DBD2
```

（2）读取或传输状态字

```
L   STW              //将状态字中的内容装入累加器 1 中
T   STW              //将累加器 1 中的内容传输到状态字中
```

(3) 装载时间值或计数值

```
LC  T1                    //将定时器 T1 中的时间值以 BCD 码格式装入累加器 1 中
L   C1                    //将计数器 C1 中的二进制格式的计数值装入累加器的低字中
```

(4) 地址寄存器装载和传输

```
LAR2   DBD 20       //将双字指针 DBD20 装入 AR2 中
LAR1   DID 30       //将数据双字指针 DID30 装入 AR1 中
LAR1   P#I 0.0      //将输入位 I0.0 的地址指针装载入 AR1 中
TAR1   AR2          //将 AR2 的内容装入 AR1 中
TAR2                //将 AR2 的内容传输至累加器 1 中
TAR1   MD24         //将 AR1 的内容传输至存储器双字 MD24 中
```

### 2. 装载和传输指令的使用

传输指令（MOVE 方块）在梯形图和功能块图中的具体应用如图 4-57a、b 所示。

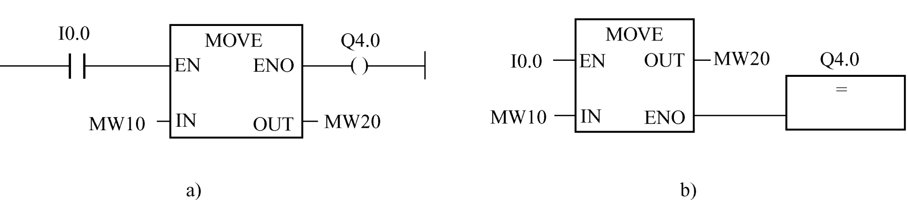



图 4-57　传输指令在梯形图和功能块图中的具体应用

说明：在传输指令中 EN 端为允许输入端；ENO 端为允许输出端。当输入 I0.0 为“1”时，传输指令将 MW10 中的字传输给 MW20。如果指令正确执行，则输出 Q4.0 为“1”。否则，如果输入 I0.0 为“0”，则数据不传输。若希望 MW10 无条件传输给 MW20，则 EN 端直接连接至母线即可。

实现上述相同功能的语句表指令为：

```
A      I0.0
JNB    _001                    //若 RLO 为“0”,则跳转到_001 处
L      MW10                    //若 RLO 为“1”,则执行装载指令
T      MW20                    //将 MW10 的内容送到 MW20 中
SET                            //置位 RLO
SAVE                           //保存 RLO
CLR                            //清除 RLO
_001: ABR                      //状态字
 =     Q4.0
```

MOVE 方块指令能传输数据长度为字节、字和双字的基本数据类型（包括常数）。如果传输的数据类型是数组或结构等，则必须用系统功能块移指令（SFC20）来实现传输。

注意：在为变量赋初始值时，为了保证传输只执行一次，一般 MOVE 方块指令和边缘触发指令联合使用。

## 4.6.2 比较指令

比较指令用于比较累加器 1 和累加器 2 中数据的大小。显然，被比较的数据必须具有相同的数据类型。比较指令可以比较整数、双整数、浮点数（实数）。比较关系包括大于、小于、等于、大于等于、小于等于、不等于几种关系。在比较条件满足的情况下，RLO 置为“1”。状态字中 CC0 和 CC1 位表示两个数的关系。

比较指令的参数类型如表 4-11 所示。

表 4-11 比较指令表

| 参数 | 数据类型 | 存储区 | 说明 |
|---|---|---|---|
| IN1，IN2 | INT<br>DINT<br>REAL | I、Q、M、D、L | 两个比较数的数据类型必须一致 |

### 1. 整数比较指令的使用

整数比较指令的梯形图如图 4-58 所示（功能块图方块指令与梯形图方块指令一致）。其中比较方块中的 I（Integer）表示整数的意思。

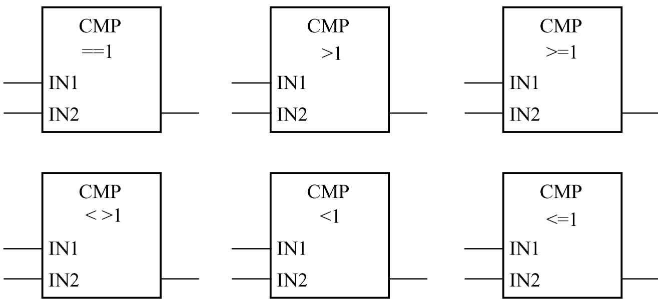



图 4-58 整数比较指令的梯形图

比较指令在梯形图和功能块图中的具体应用如图 4-59a、b 所示。

图 4-58 比较指令在梯形图和功能块图中的具体应用

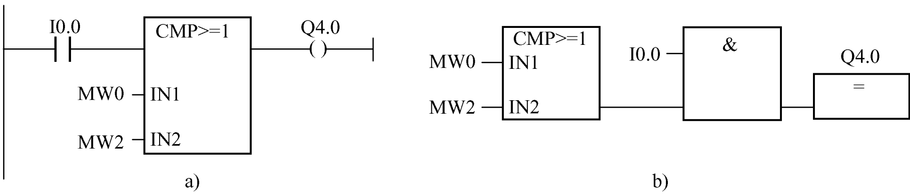



图 4-59 比较指令在梯形图和功能块图中的具体应用

说明：输入信号 I0.0 的 RLO 为“1”时，比较整数 MW0 的值是否大于等于 MW2 的值，如果是，则输出 Q4.0 为“1”。

实现上述相同功能的语句表程序为：

```
A      I0.0
A(
L      MW0              //MW0 装入累加器 1 中
L      MW2              //MW2 装入累加器 1 中,MW0 移入累加器 2 中
>=I                     //比较累加器 1 中的值是否大于等于累加器 2 中的值
)
=      Q4.0             //满足大于等于关系,则输出为 1
```

### 2. 双整数和浮点数比较指令的使用

双整数、浮点数比较指令与整数比较指令十分相似，在梯形图和功能图指令中，用 D（Double Integer）表示双整数比较，用 R（Real）表示浮点数比较。双整数比较指令的梯形图（见图4-60a），浮点数比较指令的梯形图（见图4-59b）。比较指令的梯形图和功能块图的形式一致。

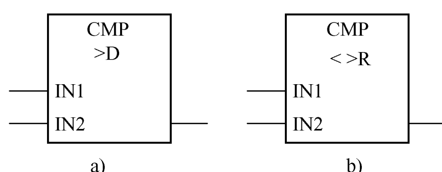



图 4-60　双整数比较指令和浮点数比较指令的梯形图

双整数比较指令在梯形图和功能块图中的具体应用如图 4-61a、b 所示。

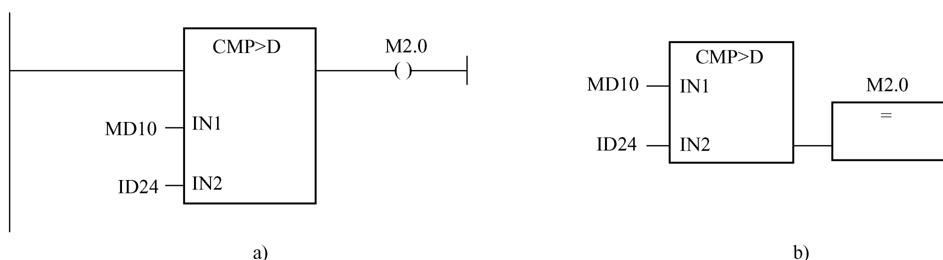



图 4-61　双整数比较指令在梯形图和功能块图中的具体应用

说明：MD10 的内容大于 ID24 的内容时，输出 M2.0 为“1”。

实现上述相同功能的语句表程序如下：

```
L   MD10        //将 MD10 双整数装入累加器 1 中
L   ID24        //将 ID24 双整数装入累加器 1 中,MD10 双整数移入累加器 2 中
>D              //累加器 2 中的值是否大于累加器 1 中的值
=   M 2.0       //如果 MD10 > ID24,则 RLO = 1,M2.0 为 1
```

注意：双整数和浮点数的操作数为双字。

**例 10**　用比较和计数指令编写开关灯程序，要求灯控按钮 I0.0 按下一次，灯 Q4.0 亮，按下两次，灯 Q4.0，Q4.1 全亮，按下三次灯全灭，如此循环。

分析：在程序中所用计数器为加法计数器，当加到 2 时，必须复位计数器，这是关键。灯控制程序如图 4-62 所示。

说明：在程序的 Network1 中，以灯控按钮 I0.0 的正跳沿触发加计数器；在 Network2 中比较可知是第一次按下按钮，所以灯 Q4.0 亮；在 Network3 中是第二次按下按钮，所以灯 Q4.1 亮；在 Network4 中是第三次按下按钮，所以灯全灭。此时使 M2.0 通电，并复位计数器。这样保证程序能顺序执行。注意，如果程序中的 M2.0 用 M0.0 取代，则很可能出错，因为 MW0 覆盖了 M0.0 的值。在编程时要特别注意这一点。

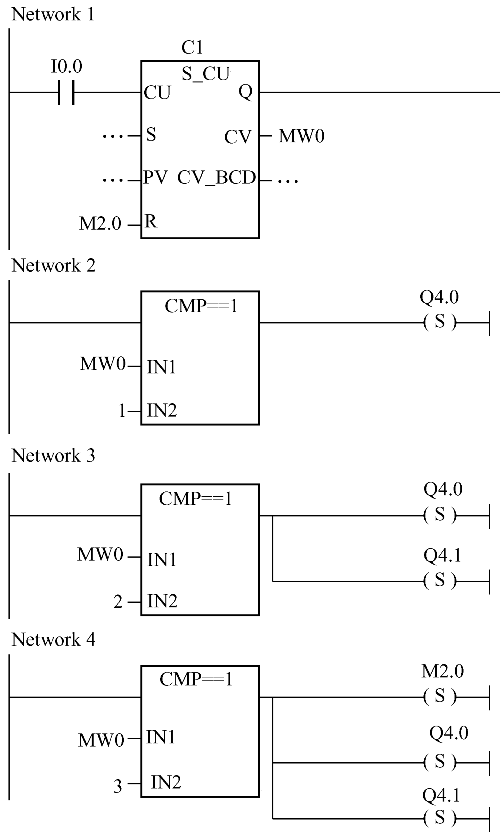



图 4-62 灯控制程序

此例如果用 Set/Reset 指令实现，则语句表程序如下：

```
Network1:按钮按下
      A    I0.0
      FP   M0.0
       = M1.0
Network2:在灯都不亮时,将 M4.1 置位
      A    M1.0
      AN   Q4.0
      AN   Q4.1
      S    M4.1                    //使 Q4.0 一个灯亮
Network3:在一个灯亮时,将 M4.2 置位
      A    M1.0
```

```
    A    Q4.0
    AN   Q4.1
    S    M4.2                    //使 Q4.0、Q4.2 两个灯亮
Network4:在两个灯全都亮时,使 M4.1 和 M4.2 复位
    A    M1.0
    A    Q4.0
    A    Q4.1
    R    M4.1
    R    M4.2                    //使两个灯一起灭。
Network5:通过 M4.1 和 M4.2,控制两盏灯的亮灭
    A    M4.1
    =Q4.0
    A    M4.2
    =Q4.1
```

注意：在 Network2 和 Network3 中不能用直接置位 Q4.0 和 Q4.1，因为循环扫描互相影响的缘故。所以在 Network5 中，通过 M4.1 和 M4.2 来控制输出。

**例 11**　如图 4-63 所示为仓库区及显示面板。在两个传送带之间有一个装 100 件物品的仓库，传送带 1 将物品送至临时仓库。传送带 1 靠近仓库区一端的光电传感器（I0.0）确定有多少物品运送至仓库区，传送带 2 将仓库区中的物品运送至货场，传送带 2 靠近仓库区一端的光电传感器（I0.1）确定已有多少物品从库区送至货场。显示面板上有五个指示灯（Q12.0～Q12.4）显示仓库区物品的占有程度。

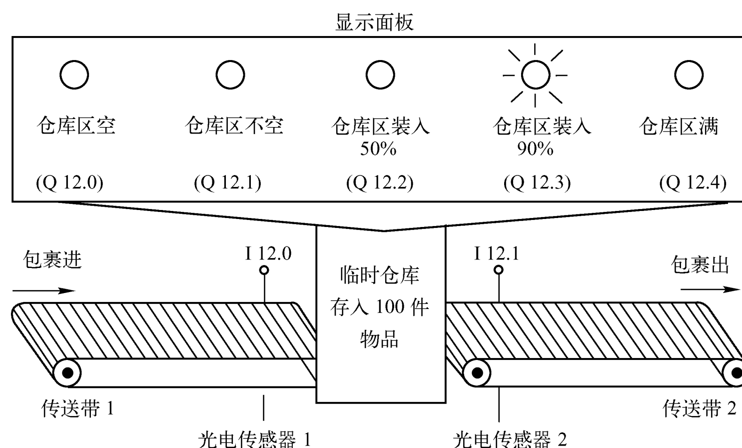



图 4-63　仓库区及显示面板

图 4-64 给出了显示面板上指示灯控制程序。

分析：输入 I0.0 信号每次从“0”变到“1”时，计数器加 1，表示光电传感器检测到物品进入仓库；当输入 I0.1 信号每次从“0”变到“1”时，计数器减 1，表示光电传感器检测到物品送出仓库。其中显示 50% 的指示时，是物品数在 50%～90% 范围时的显示。所以运用大于等于比较器和小于比较器的并联。同样物品数在 90%～100% 范围时，也运用两个比较器实现范围的限定。

说明：在程序的 Network1 中，通过光电传感器 I0.0 的正跳沿脉冲触发加计数器，光

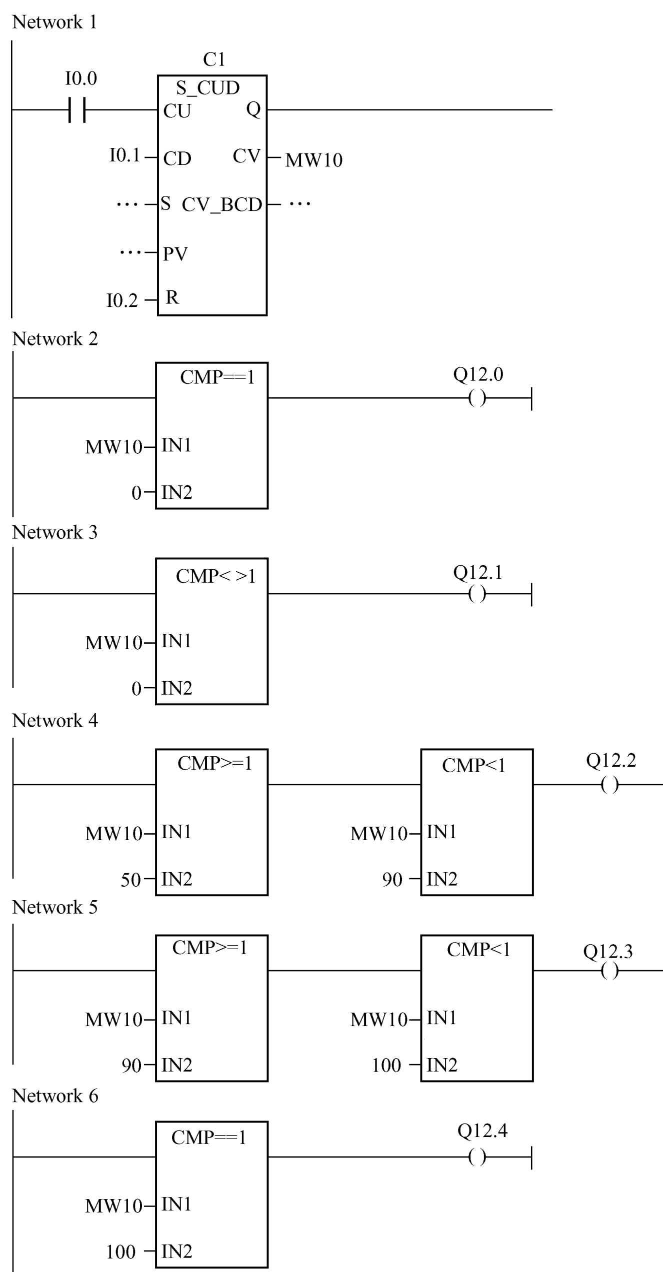



图 4-64　显示面板上指示灯控制程序

电传感器 I0. 1 的正跳沿脉冲触发减计数器，当前计数值在 MW10 中显示；在 Network2 和 Network3 中，通过与 0 的比较指令可知，仓库区的状态为空还是非空，分别用指示灯 Q12. 0 和 Q12. 1 表示；在 Network4 和 Network5 中分别通过两个比较器的并联可知仓库区的状态在 50% ~90% 之间，或是在 90% ~100% 之间，两种状态用指示灯 Q12. 2 和 Q12. 3 表示；在 Network6 中显示仓库区满载，这一状态用 Q12. 4 表示。计数器复位用按钮 I0. 2 来实现。

对应的语句表程序如下：

```
Network1
    A    I 0.0         //在 I0.0 的正跳沿
    CU   C1            //加 1
    A    I 0.1         //在 I0.1 的正跳沿
    CD   C1            //减 1
    A    I0.2
    R    C1            //用 I0.2 复位 C1
Network2
    L    C1
    T    MW10          // 将 C1 当前值存入 MW10 中
    L    MW10
    L    0
    ==I
    =    Q12.0         //在等于 0 时,使仓库区空显示灯亮
Network3
    L    MW10
    L    0
    <>I
    =    Q12.1         //不等于 0 时,使仓库区不空显示灯亮
Network4
    A(
    L    MW10
    L    50
    >=I
    )
    A(
    L    MW10
    L    90
    <I
    )
    =    Q12.2         //在 50% 以上时,使仓库区装入 50% 指示灯亮
Network5
    A(
    L    MW10
    L    90
    >=I
    )
    A(
    L    MW10
    L    100
    <I
```

```
    )
    =    Q12.3              //在 90% 以上时,使仓库区装入 90% 指示灯亮
Network6
    L    MW10
    L    100
    ==I
    =    Q12.4              //在 100% 时,使仓库区满指示灯亮
```

## 4.6.3 转换指令

转换指令将源数据按照规定的格式读入累加器，在累加器中对数据进行类型转换，再将转换后的结果传送到目的地址。转换操作有：BCD 码和整数及长整数间的转换，浮点数和长整数间的转换，数的取反等。数据转换指令如表 4–12 所示。

**表 4–12　数据转换指令**

| 梯形图 | 功能块图 | 语句表 | 说　　明 |
|---|---|---|---|
| BCD_I<br>EN ENO<br>IN OUT | BCD_I<br>EN OUT<br>IN ENO | BTI | 将累加器 1 中的 3 位 BCD 码转换成整数 |
| I_BCD<br>EN ENO<br>IN OUT | I_BCD<br>EN OUT<br>IN ENO | ITB | 将累加器 1 中的整数转换成 3 位 BCD 码 |
| BCD_DI<br>EN ENO<br>IN OUT | BCD_DI<br>EN OUT<br>IN ENO | BTD | 将累加器 1 中的 7 位 BCD 码转换成双整数 |
| DI_BCD<br>EN ENO<br>IN OUT | DI_BCD<br>EN OUT<br>IN ENO | DTB | 将累加器 1 中的双整数转换成 7 位 BCD 码 |
| DI_R<br>EN ENO<br>IN OUT | DI_R<br>EN OUT<br>IN ENO | DTR | 将累加器 1 中的双整数转换成浮点数 |
| I_DI<br>EN ENO<br>IN OUT | I_DI<br>EN OUT<br>IN ENO | ITD | 将累加器 1 中的整数转换成双整数 |
| ROUND<br>EN ENO<br>IN OUT | ROUND<br>EN OUT<br>IN ENO | RND | 将浮点数转换成四舍五入的双整数 |
| CEIL<br>EN ENO<br>IN OUT | CEIL<br>EN OUT<br>IN ENO | RND + | 将浮点数转换成大于等于它的最小双整数 |

（续）

| 梯形图 | 功能块图 | 语句表 | 说　　明 |
|---|---|---|---|
| FLOOR<br>EN ENO<br>IN OUT | FLOOR<br>EN OUT<br>IN ENO | RND − | 将浮点数转换成小于等于它的最大双整数 |
| TRUNC<br>EN ENO<br>IN OUT | TRUNC<br>EN OUT<br>IN ENO | TRUNC | 将浮点数转换成截位取整的双整数 |
| | | CAW<br>CAD | 交换累加器 1 低字中 2 个字节的位置<br>交换累加器 1 中 4 个字节的顺序 |

**1. BCD 码的数据格式**

BCD 码数据格式有两种：一种是字（16 位）格式的 BCD 码数，其中第 0～11 位用来表示 3 位 BCD 码，每 4 位二进制数用来表示一位 BCD 码，第 15 位用来表示 BCD 码的符号，正数为 0，负数为 1，第 12～14 位未用。16 位 BCD 码取值范围是 −999～+999；另一种是双字（32 位）格式的 BCD 码数，其中第 0～27 位用来表示 7 位 BCD 码，每 4 位二进制数用来表示一位 BCD 码，第 31 位用来表示 BCD 码的符号，正数为 0，负数为 1，第 12～14 位未用。32 位 BCD 码范围是 −9 999 999～+9 999 999。

**2. BCD 码和整数间的转换**

在执行 ITB 指令时，由于 3 位 BCD 码的范围是 −999～+999，小于 16 位整数的数值范围，因此一个整数到 BCD 码的转换并不总是可行的，如果整数超出了 BCD 码的范围，将得不到有效的转换结果。同时状态字中溢出位（OV）和溢出保持位（OS）置 1。程序中，需要根据状态位 OV 或 OS 判断结果是否有效。同样，在执行 DTB 指令时也会有相同情况出现。如果 BCD 码是无效数，程序将被终止，并有以下相关事件发生：

1）CPU 进入 STOP 状态。“BCD 码转换错误”写入诊断缓冲区。

2）如果 OB121 已编程就调用之（OB121 的含义见第 8 章）。

BCD 码转换为双整数的梯形图如图 4-65 所示。

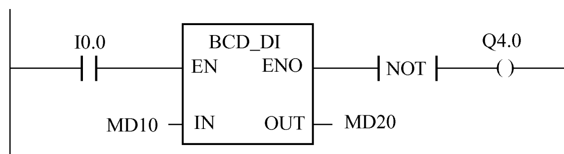



图 4-65　BCD 码转换为双整数的梯形图

说明：在起动转换信号 I0.0 有效时，将存于 MD10 中的 BCD 码转换为双整数，并存于 MD20 中。若转换没有执行，则输出位 Q4.0 为“1”。

对应的语句表程序如下：

```
A(
AI0.0
```

```
        JNB_001
        LMD10
        BTD                //将 BCD 码(MD10)转换成双整数
        TMD20
        SET
        SAVE
        CLR
_001：ABR
        )
        NOT
        = Q4.0             //转换不成功,则输出 Q4.0 为 1
```

执行 BTD 指令，将 BCD 码 MD10 转换为双整数 MD20，如图 4-66 所示。

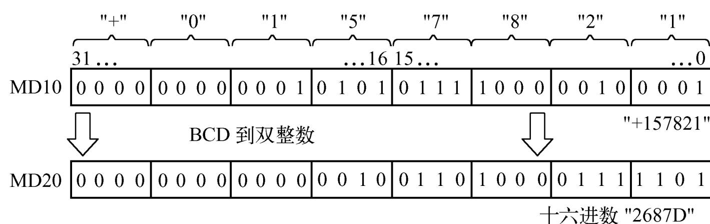



图 4-66　BCD 码 MD10 转换为双整数 MD20

其余转换大同小异，具体见表 4-12。

**3. 浮点数和双整数之间的转换**

在 RND 指令中，因为浮点数的数值范围远大于 32 位整数，所以有些浮点数不能成功转换为 32 位整数。非法操作将使状态字中的 OV 和 OS 位被置 1。

各种的取整指令要依据用户和精度的要求具体使用。

**例 12**　将 101 英寸转换为以厘米为单位的整数，并送至 MW30 中。

STL 程序如下：

```
L   101            //装载 16 位常数
ITD                //转换为 32 位双整数
DTR                //转换为浮点数
L   2.54           //装载浮点数(转换比率)
*R                 //将英寸进行转换
RND                //取整
T   MW30           //传输到 MW30 中
```

**4. 取反与求补指令**

取反和求补指令如表 4-13 所示。

**5. 交换累加器 1 中字节的位置**

CAW 指令交换累加器 1 低字中两个字节的位置，累加器 1 中的高字不变。

CAD 指令交换累加器 1 中四个字节的位置，交换后的累加器内容如表 4-14 所示。

表 4-13　取反和求补指令表

| 梯形图 | 功能块图 | 语句表 | 说　明 |
|---|---|---|---|
| INV_I<br>EN ENO<br>IN OUT | INV_I<br>EN OUT<br>IN ENO | INVI | 将累加器 1 低字中的 16 位整数取反码 |
| INV_DI<br>EN ENO<br>IN OUT | INV_DI<br>EN OUT<br>IN ENO | INVD | 将累加器 1 的双整数取反码 |
| NEG_I<br>EN ENO<br>IN OUT | NEG_I<br>EN OUT<br>IN ENO | NEGI | 将累加器 1 低字中的 16 位整数取补码 |
| NEG_DI<br>EN ENO<br>IN OUT | NEG_DI<br>EN OUT<br>IN ENO | NEGD | 将累加器 1 的双整数取补码 |
| NEG_R<br>EN ENO<br>IN OUT | NEG_R<br>EN OUT<br>IN ENO | NEGR | 将累加器 1 中浮点数的符号位取反 |

表 4-14　CAD 指令功能（交换后累加器内容）

| 累加器的字节 | ACCU1 – HH | ACCU1 – HL | ACCU1 – LH | ACCU1 – LL |
|---|---|---|---|---|
| 交换前内容 | A | B | C | D |
| 交换后内容 | D | C | B | A |

### 4.6.4　移位和循环移位指令

移位指令是将输入端 IN 中的内容向左或向右移动。移动位数由输入值 N 确定。移位后空出的位填以 0 或者符号位。被移动的最后一位保存在状态字 CC1 中，用户可以使用条件跳转指令对 CC1 进行判断。

循环指令与一般的移位指令有所不同，它不仅移位，而且移位后空出的位填入从 IN 中移出的位，形成一个封闭的环。

在 S7 – 300 中，若移出（OUT）和移入（IN）为同一存储字或双字时，即连续移位，则移位及循环移位指令必须与边缘触发指令结合使用，确保在使能输入信号 EN 的一个扫描周期内移位一次。否则，在输入信号 EN 的高电平期间，移位次数不受控制。这种移位及循环移位指令为微分类型移位。

**1．无符号移位指令**

如表 4-15 所示为无符号数的移位指令的格式。

如图 4-67 所示为字的左移（6 位）指令，如图 4-68 所示为双字的右移（3 位）指令。

表 4-15 无符号数的移位指令

| 梯形图 | 功能块图 | 语句表 | 说　　明 |
|---|---|---|---|
| SHL_W<br>EN ENO<br>IN OUT<br>N | SHL_W<br>EN<br>IN OUT<br>N ENO | SLW | 将 IN 中的字逐位左移，空出位填 0 |
| SHR_W<br>EN ENO<br>IN OUT<br>N | SHR_W<br>EN<br>IN OUT<br>N ENO | SRW | 将 IN 中的字逐位右移，空出位填 0 |
| SHL_DW<br>EN ENO<br>IN OUT<br>N | SHL_DW<br>EN<br>IN OUT<br>N ENO | SHL_DW | 将 IN 中的双字逐位左移，空出位填 0 |
| SHR_DW<br>EN ENO<br>IN OUT<br>N | SHR_DW<br>EN<br>IN OUT<br>N ENO | SHR_DW | 将 IN 中的双字逐位右移，空出位填 0 |

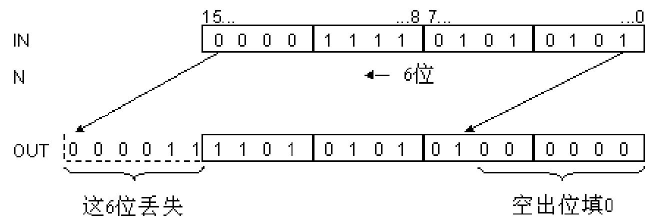



图 4-67 字的左移（6 位）指令

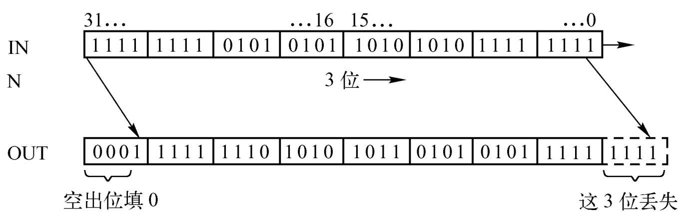



图 4-68 双字的右移（3 位）指令

双字左移指令在梯形图和功能块图中的具体应用如图 4-69a、b 所示。

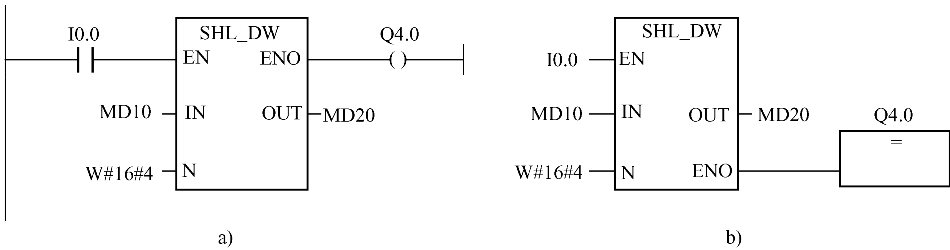



图 4-69 双字左移指令在梯形图和功能块图中的具体应用

说明：当 I0.0 接通时，双字左移指令开始工作，将 MD10 中的内容左移 4 位，并将结果存入 MD20 中，如果移位指令执行，则输出 Q4.0 为“1”。

实现上述相同功能的语句表程序如下：

```
      A     I0.0
      JNB   _001
      L     W#16#4
      L     MD10
      SLD                  //双字左移 4 位
      T     MD20           //将移位结果存入 MD20 中
      SET
      SAVE
      CLR
_001：A     BR
      =     Q4.0           //如果移位指令执行,则输出 Q4.0 为“1”
```

### 2. 有符号移位指令

如表 4-16 所示为有符号数右移指令的格式。

**表 4-16　有符号数右移指令表**

| 梯形图 | 功能块图 | 语句表 | 说　明 |
|---|---|---|---|
| SHR_I<br>EN　ENO<br>IN　OUT<br>N | SHR_I<br>EN<br>IN　OUT<br>N　ENO | SSI | 将 IN 中的字逐位右移，空出位填以符号位（正数填 0，负数填 1） |
| SHR_DI<br>EN　ENO<br>IN　OUT<br>N | SHR_DI<br>EN<br>IN　OUT<br>N　ENO | SSD | 将 IN 中的双字逐位右移，空出位填以符号位（正数填 0，负数填 1） |

如图 4-70 所示为有符号字的右移（4 位）。

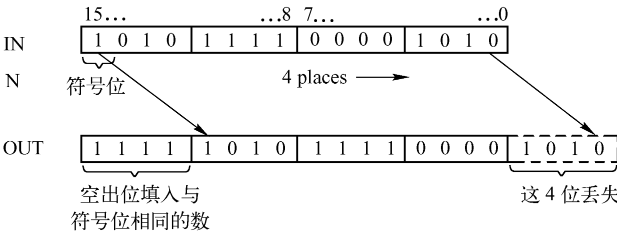



图 4-70　有符号字的右移（4 位）

有符号移位指令的具体用法与无符号移位指令相似，这里就不再举例。

### 3. 循环移位指令

如表 4-17 所示为循环移位指令的格式。

表 4-17　循环移位指令表

| 梯形图 | 功能块图 | 语句表 | 说　明 |
|---|---|---|---|
| ROL_DW<br>EN ENO<br>IN OUT<br>N | ROL_DW<br>EN<br>IN OUT<br>N ENO | RLD | 将 IN 中的双字逐位左移，空出位填以移出位 |
| ROR_DW<br>EN ENO<br>IN OUT<br>N | ROR_DW<br>EN<br>IN OUT<br>N ENO | RRD | 将 IN 中的双字逐位右移，空出位填以移出位 |

如图 4-71 所示为循环右移（3 位）。

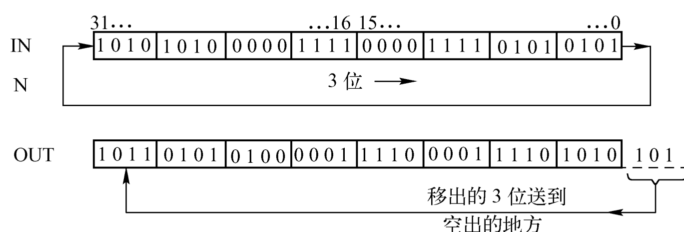



图 4-71　循环右移（3 位）

循环移位指令的具体用法与无符号移位指令相似，这里不再举例。

## 4.6.5　累加器操作和地址寄存器指令

### 1. 累加器操作指令

累加器操作指令见表 4-18。

表 4-18　累加器操作指令

| 指　令 | 说　明 |
|---|---|
| TAK | 将累加器 1 和累加器 2 的内容交换 |
| PUSH | 压栈 |
| POP | 出栈 |
| INC | 将累加器 1 低字节内容加指定常数值，指令执行是无条件的，且不影响状态字 |
| DEC | 将累加器 1 低字节内容减指定常数值，指令执行是无条件的，且不影响状态字 |

S7-300 的 CPU 中有 2 个累加器，而 S7-400 中则有 4 个累加器。累加器组成一个堆栈，堆栈的数据按照“先进后出”的原则进行数据的存储。PUSH 指令将堆栈中的各层数据依次下压，最低层数据丢失。如果有 4 个累加器，则累加器 3 中的内容复制到累加器 4 中，累加器 2 中的内容复制到累加器 3 中，以此类推。出栈过程则相反。

在编程中，循环执行某段程序十分常见，而 INC 指令在循环中起比较重要的作用。下面说明 INC 的用法。

**例 13**　完成从 1 ~5 共 5 个数的叠加。以 MB10 为变量进行循环处理，以 MW30 存储累加和，结果送入 MW40 中。

STL 程序如下：

```
        L      0
        T      MW30   //给累加和 MW30 赋初值
        L      1
        T      MB10   //给累加变量 MB10 赋初值
Label1：L      MW30
        L      MB10
        +I            //MW30 与 MB10 相加
        T      MW30   //结果送入 MW30 中
        L      MB10
        INC    1      //变量 MB10,加 1
        T      MB10
        L      MB10
        L      B#16#5 //MB10 与常数 5 比较
        <=I
        JC     Label1 //小于等于 5 则循环跳转至 Label1 处
        L      MW30   //否则,将累加和结果送入 MW40 中
        T      MW40
```

说明：通过比较指令，条件满足则有条件跳转至 Label1 处，否则，循环结束，将结果输出。

TAK 指令是将累加器中的内容进行交换。具体用法如下：

**例 14**　两个数分别装在 MW10 和 MW12 中，试编程实现大数减小数的功能，结果传输到 MW14 中。

```
       L    MW10       //将 MW10 装入累加器 1 中
       L    MW12       //将 MW12 装入累加器 1 中,MW10 移入累加器 2 中
       >I              //MW10 中的数值 >MW12 的数值?
       JC   NEXT       //大于,跳转至 NEXT 标号处
       TAK             //否则,交换累加器中数值
NEXT:—I                //大数减小数
       T    MW14       //结果传送给 MW14
```

**2. 地址寄存器指令**

地址寄存器指令如表 4-19 所示。

**表 4-19　地址寄存器指令**

| 指令 | 操作数 | 说　明 |
|---|---|---|
| +AR1 | 无 | 指令没有指明操作数，则把累加器 1 中低字的内容加至地址寄存器 1 |
| +AR2 | 无 | 指令没有指明操作数，则把累加器 1 中低字的内容加至地址寄存器 2 |
| +AR1 | P#Byte. Bit | 把一个指针常数加至地址寄存器 1 |
| +AR2 | P#Byte. Bit | 把一个指针常数加至地址寄存器 2 |

+AR1 指令将地址寄存器 AR1 的内容加上作为地址偏移量的累加器 1 中低字的内容，或加上指令中的 16 位常数，结果存在 AR1 中。

+AR2 指令将地址寄存器 AR2 的内容加上作为地址偏移量的累加器 1 中低字的内容，或加上指令中的 16 位常数，结果存在 AR2 中。

注意：在没有指明操作数的地址寄存器指令中，若累加器中的内容为 16 位有符号数，则首先应将其扩充为 24 位数，并且符号保持不变，之后与地址寄存器中的低 24 位有效数字相加，而地址寄存器中的存储区域标识符（第 24~26 位）保持不变。

在使用地址寄存器加指令时，应保证累加器 1 或指针常数的正确格式。其语句表的具体用法如下：

```
L      +300          //将常数 300 装入累加器 1 中
+AR1                 //与累加器 1 中低字的内容相加,结果送到 AR1 中
+AR1P#300.0          //AR1 的内容加地址偏量,结果送入 AR1
```

### 3. 数据块指令

数据块指令如表 4-20 所示。

表 4-20　数据块指令

| 梯形图 | 功能图 | 语句表指令 | 说明 |
|---|---|---|---|
| —(OPN) | OPN | OPN | 打开数据块作为共享或背景数据块 |
| | | CAD | 交换数据块寄存器 |
| | | DBLG | 将共享数据块的长度装入累加器 1 中 |
| | | DBNO | 将共享数据块的块号装入累加器 1 中 |
| | | DILG | 将背景数据块的长度装入累加器 1 中 |
| | | DINO | 将背景数据块的块号装入累加器 1 中 |

使用上述指令必须先用 OPN 指令打开一个数据块，之后才能使用其他的数据块指令。数据块指令在梯形图和功能块图中的具体用法如图 4-72a、b 所示。

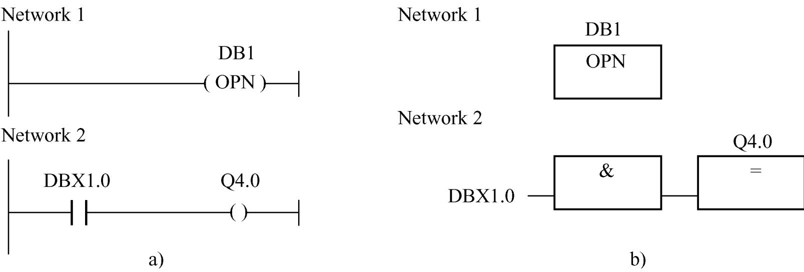



图 4-72　数据块指令在梯形图和功能块图中的具体应用

说明：在 Network1 中打开数据块 DB1；在 Network2 中，用数据块 DB1 中的 DBX1.0 控制 Q4.0 的通断。

实现上述相同功能的语句表指令如下：

```
OPN   DB1            //打开数据块 DB1
```

```
A      DBX1.0
=      Q4.0          //若 DBX1.0 的 RLO 为 1,则输出 Q4.0 为 1
```

#### 4. 显示和空操作指令

显示和空操作指令如表 4-21 所示。这些指令是在梯形图转换为语句表指令时自动添加的，无特别的意义。但在语句表中将这些指令去掉，语句表指令就不能再转换为梯形图。

**表 4-21　显示和空操作指令**

| STL 指令 | 说　　明 |
|---|---|
| BLD | 显示程序的形式，执行该指令不产生任何影响 |
| NOP 0 | 空操作 |
| NOP 1 | 空操作 |

## 4.7　数据运算指令

算术运算十分重要，因为一般的自动控制系统都需要 PID 控制器，其算法的实现离不开基本的算术运算。在 S7 中可以对整数、长整数和浮点数进行基本算术运算。这些指令是在累加器 1 和累加器 2 中进行的。其中，累加器 2 中的值作为被减数或被除数，累加器 1 则作为减数和除数，算术运算结果保存在累加器 1 中。在进行算术运算时，不必考虑对 RLO 的影响。需要考虑的是状态字的 CC0、CC1、OS、OV 位。具体操作中运用位操作指令或条件跳转指令进行判断。

### 4.7.1　整数算术运算

整数算术运算如表 4-22 所示。

**表 4-22　整数算术运算**

| 梯　形　图 | 功 能 块 图 | 语句表指令 | 类 型 说 明 |
|---|---|---|---|
| ADD_I<br>EN ENO<br>IN1<br>IN2 OUT | ADD_I<br>EN<br>IN1 OUT<br>IN2 ENO | +I | 将 IN1 和 IN2 中的 16 位整数相加 |
| | | -I | 将 IN1 中的 16 位整数减去 IN2 中的 16 位整数 |
| | | *I | 将 IN1 和 IN2 中的 16 位整数相乘，结果以 32 位整数输出 |
| | | /I | 将 IN1 中的 16 位整数除以 IN2 中的 16 位整数 |
| MUL_DI<br>EN ENO<br>IN1<br>IN2 OUT | MUL_DI<br>EN<br>IN1 OUT<br>IN2 ENO | +D | 将 IN1 和 IN2 中的 32 位整数相加 |
| | | -D | 将 IN1 中的 32 位整数减去 IN2 中的 32 位整数 |
| | | *D | 将 IN1 和 IN2 中的 32 位整数相乘结果以 32 位整数输出 |
| | | /D | 将 IN1 中的 32 位整数除以 IN2 中的 32 位整数，商输出 |
| | | MOD | 将 IN1 中的 32 位整数除以 IN2 中的 32 位整数，余数输出 |

表 4-22 中没有完全列出梯形图和功能块图的所有整数算术运算方块指令，其余的方块指令除了方块中的标识不同外，输入、输出格式相同，通过标识用户可以很容易地判断出整数运算的类型。例如，SUB 和 DIV 分别为减法指令和除法指令。

在 16 位整数乘法运算中，运算结果为 32 位双整数，并存入累加器 1 中。如果运算后状

态字的 OS 和 OV 位均为 1，表示运算结果超出了 16 位整数允许的范围。

在 16 位整数除法运算中，16 位商存在累加器 1 的低字中，余数在累加器 1 的高字中。

在 32 位双整数乘法运算中，运算结果为 32 位双整数，并存入累加器 1 中。如果运算后状态字的 OS 和 OV 位均为 1，表示运算结果超出了 32 位整数允许的范围。

在 32 位整数除法运算中，32 位商存在累加器 1 中，余数被丢掉。

在梯形图指令中，若运算结果超出允许范围，OS 和 OV 位均为 1，输出为 0。

**例 15** 用语句表实现字运算 MW4 + MW6 - 2 的程序，其运算结果送入 MW10 中。

```
L   MW4       //将 MW4 装入累加器 1 中
L   MW6       //将 MW6 装入累加器 1 中,MW4 移入累加器 2
+I            //相加
+   L# - 2    //减常数 2
T   MW10      //结果存入 MW10 中
```

**例 16** 用梯形图实现运算（10000 * MD6）/27666，结果存入 MW10 中。

双整数运算梯形图实现如图 4-73 所示。

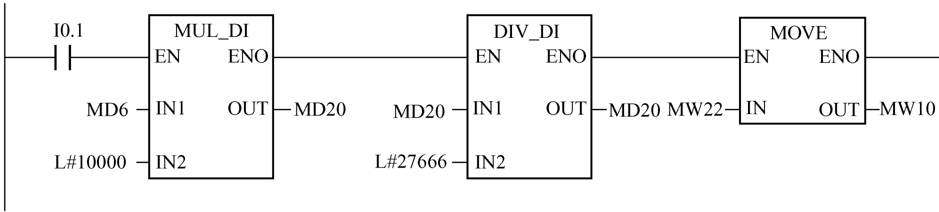



图 4-73 双整数运算梯形图

说明：在 I0.1 得电时，才能进行运算。图中因为方块的串联，所以相互之间有嵌套关系。

注意：在运算中，如果存放于存储器中的数乘以 10000 后，可能超出 16 位整数范围，这时如果用梯形图中的 MUL_ I 指令，运算结果为 16 位整数，明显不适合，所以需要使用双字乘法指令 MUL_ DI。在进行除法运算时，虽然双字除法的运算结果仍然为双字，但此题中实际的运算结果没有超出 16 位整数的最大值，所以结果通过 MOVE 指令，只保留低字 MW22 中 16 位运算结果。

## 4.7.2 浮点数算术运算

浮点数数学运算指令是对累加器 1 和累加器 2 中的 32 位 IEEE 格式的浮点数进行运算，运算结果存在累加器 1 中。浮点数算术运算如表 4-23 所示。

**表 4-23 浮点数算术运算**

| 梯 形 图 | 功 能 块 图 | 语句表指令 | 类 型 说 明 |
|---|---|---|---|
| ADD_R<br>EN ENO<br>IN1<br>IN2 OUT | ADD_R<br>EN<br>IN1 OUT<br>IN2 ENO | +R | 将 IN1 和 IN2 中的浮点数相加 |
| | | -R | 将 IN1 中的浮点数减去 IN2 中的浮点数 |
| | | *R | 将 IN1 和 IN2 中的浮点数相乘 |
| | | /R | 将 IN1 中的浮点数除以 IN2 中的浮点数 |

（续）

| 梯　形　图 | 功 能 块 图 | 语句表指令 | 类 型 说 明 |
|---|---|---|---|
| SQR<br>EN ENO<br>IN OUT | SQRT<br>EN OUT<br>IN ENO | ABS | 将输入 IN 的浮点数值取绝对值 |
| | | SQR | 将输入 IN 的浮点数值取平方值 |
| | | SQRT | 将输入 IN 的浮点数值取平方根值 |
| | | LN | 将输入 IN 的浮点数值取自然对数 |
| | | EXP | 将输入 IN 的浮点数值取指数值 |
| COS<br>EN ENO<br>IN OUT | COS<br>EN OUT<br>IN ENO | SIN | 求输入 IN 中以弧度表示的角度值的正弦值 |
| | | COS | 求输入 IN 中以弧度表示的角度值的余弦值 |
| | | ASIN | 求输入 IN 中浮点数的反正弦值 |
| | | ACOS | 求输入 IN 中浮点数的反余弦值 |
| | | TAN | 求输入 IN 中以弧度表示的角度值的正切值 |
| | | ATAN | 求输入 IN 中以弧度表示的角度值的余切值 |

**例 17**　浮点数平方指令的使用。求存于 DB10. DBD0 的平方，并且将结果存于 DB10. DBD4 中。

语句表程序如下：

```
      OPN   DB10          //打开数据块 DB10
      L   DBD0            //装载浮点数 DB10 中的 DBD0 于累加器 1 中
      SQR                 //求平方
      AN OV               //如果运算正确
      JC OK               //转至 OK 处
      BEU                 //否则,无条件停止
OK：TD   BD4              //正确结果传输到 DBD4 中
```

**例 18**　浮点数对数指令和指数指令的用法。计算 EXP（3 * LN（7）），并将结果存入 MW40。

STL 语句表程序如下：

```
L   L#5        //装载 32 位整数常数 5 于累加器 1 中
DTR            //将其转换为浮点数
LN             //取对数
L   3.0        //装载 32 位浮点数 3.0 于累加器 1 中,对数运算结果存于累加器 2 中
 * R           //与 3.0 进行浮点数相乘
EXP            //取指数
RND            //将浮点数结果转换为双整数
T   MW40       //存于 MW40 中
```

### 4.7.3　字逻辑运算指令

字逻辑运算指令如表 4-24 所示。字逻辑运算结果存在累加器 1 中。

表 4-24 字逻辑运算指令

| 梯 形 图 | 功 能 块 图 | 语句表指令 | 操作类型说明 |
|---|---|---|---|
| WOR W: EN, IN1, IN2 → ENO, OUT | WOR_W: EN, IN1, IN2 → OUT, ENO | AW | 将 IN1 和 IN2 中的字相与 |
| | | OW | 将 IN1 和 IN2 中的字相或 |
| | | XOR | 将 IN1 和 IN2 中的字相异或 |
| WAND_DW: EN, IN1, IN2 → ENO, OUT | WAND_DW: EN, IN1, IN2 → OUT, ENO | AD | 将 IN1 和 IN2 中的双字相与 |
| | | OD | 将 IN1 和 IN2 中的双字相或 |
| | | XOD | 将 IN1 和 IN2 中的双字相异或 |

**例 19** 字或指令的使用。要求：取出 QW10 中低 4 位值，并重新存于 QW10 中。

STL 语句表程序如下：

```
L   QW10          //装 QW10 于累加器 1 中
L   W#16#000F     //装 W#16#000F 于累加器 1 中,原累加器 1 的内容移至累加器 2 中
AW                //累加器 1 中低字内容与 W#16#000F 相与
T   QW10          //取出低 4 位结果存于 QW10 中
```

**例 20** 运用字逻辑运算指令对输入 I12.0～I13.7 共 16 位输入进行跳变沿检测。

首先进行正跳沿检测，语句表程序如下：

```
L   MW10          //将输入位的上一个周期状态装入累加器 1 中
L   IW12          //将输入位当前状态装入累加器 1 中,上一周期状态移入累加器 2 中
T   MW10          //保存当前状态供下一周期使用
XOW               //异或操作,当前状态与前不同的位被置 1
L   MW10          //重新装入当前状态
AW                //与操作,当前状态为 0 的位被清零(负跳变被屏蔽)
T   MW14          //将正跳沿结果送入 MW14 中
```

再进行负跳沿检测，语句表程序如下：

```
L   MW12          //将输入位的上一个周期状态装入累加器 1 中
L   IW12          //将输入位当前状态装入累加器 1 中,上一周期状态移入累加器 2 中
T   MW12          //保存当前状态供下一周期使用
XOW               //异或操作,当前状态与前不同的位被置 1
L   MW12          //重新装入当前状态
INVI              //将当前状态取反
AW                //与操作,当前状态为 1 的位被清零(正跳变被屏蔽)
T   MW14          //将负跳变结果送入 MW14 中
```

## 4.8 控制指令

控制指令控制程序的执行顺序，使得 CPU 能根据不同的情况，执行不同的指令。控制指令有两种：逻辑控制指令和程序控制指令。

### 4.8.1 逻辑控制指令

逻辑控制指令是指逻辑块中的跳转和循环指令。在没有执行跳转和循环指令之前，各语句按先后顺序执行，这种执行方式称为线性扫描。而逻辑控制指令终止了线性扫描，跳转到地址标号（Label）所指定的目的地址。之后，程序再次开始线性扫描。这里需注意几点：跳转指令不执行跳转指令和标号之间的程序；跳转可以是从上至下，也可以反其道而行之；跳转指令只能在同一逻辑块内跳转，而不能在不同逻辑块之间跳转；在同一块中，跳转目的地址只能出现一次，否则，程序将不知道究竟往哪里跳转。

跳转和循环指令的操作数是地址标号，标号最多有四个字符。由于标号是指目的地址，所以又称为目的地址标号。在语句表中，目的标号与目的指令之间用“:”分隔，而在梯形图中目的地址标号必须在一个网络的开始。

跳转指令有几种形式：无条件跳转指令、多分支跳转指令、与 RLO 和 BR 有关的跳转指令、与信号状态位有关的跳转指令、与条件码 CC0 和 CC1 有关的跳转指令。

逻辑控制指令如表 4-25 所示。

表 4-25　逻辑控制指令

| 指令 | 说　明 | 指令 | 说　明 |
|---|---|---|---|
| JU | 无条件跳转 | JOS | OS = 1 时跳转 |
| JL | 多分支跳转 | JZ | 累加器 1 中计算结果为零时跳转 |
| JC | 当 RLO = 1 时，跳转 | JN | 累加器 1 中计算结果非零时跳转 |
| JCN | 当 RLO = 0 时，跳转 | JP | 累加器 1 中计算结果为正时跳转 |
| JCB | 当 RLO = 1，且 BR = 1 时跳转 | JM | 累加器 1 中计算结果为负时跳转 |
| JNB | 当 RLO = 0，且 BR = 1 时跳转 | JMZ | 累加器 1 中计算结果小于等于零时（非正）跳转 |
| JBI | BR = 1 时跳转 | JPZ | 累加器 1 中计算结果大于等于零时（非负）跳转 |
| JNBI | BR = 0 时跳转 | JUO | 浮点数溢出跳转 |
| JO | OV = 1 时跳转 | LOOP | 循环指令 |

**1. 无条件跳转指令**

无条件跳转指令（Jump Unconditional，JU）无条件地中断正常程序的线性扫描，跳转到目标地址标号所在的程序段处继续执行。无条件跳转与状态字的状态无关。

**2. 多分支跳转指令**

多分支跳转的特点是程序从一个公共点起开始分支，在各分支执行各自相关程序之后，又汇总到另一个公共点。这在编程中会经常使用。多分支跳转指令（Jump Via Jump to List，JL）必须与无条件跳转指令 JU 一起使用。多分支的跳步目标号在累加器 1 的低字节中，且最多有 255 个分支，每个跳转目标由一条 JU 指令和一个标号组成。

下面介绍多分支跳转指令及无条件跳转指令的具体使用，其 STL 程序如下：

```
L   MB0      //将跳转目标号装载到累加器 1 的低字节中
JL  ERR      //如果累加器 1 低字节内容 >3,则跳转至标号 ERR 处
JU  SEG0     //如果累加器 1 低字节内容 =0,则跳转至标号 SEG0 处
JU  SEG1     //如果累加器 1 低字节内容 =1,则跳转至标号 SEG1 处
```

```
        JU  SEG2         //如果累加器 1 低字节内容 =2,则跳转至标号 SEG2 处
        JU  SEG3         //如果累加器 1 低字节内容 =3,则跳转至标号 SEG3 处
ERR:    JU  COMM
SEG0:   … .              //程序段 0
        … .
        JU  COMM
SEG1:   … .              //程序段 1
        … ..
        JU  COMM
SEG2:   … .              //程序段 2
        … ..
        JU  COMM
SEG3:   … .              //程序段 3
        … ..
        JU  COMM
COMM:   … ..
```

说明：从上面的语句表程序中可以看出：在 MB0 中装载了分支的跳转目标号（0 ~ n），本例中 JU 指令有 4 条（0 ~ 3），包括错误标识 ERR（JL 语句），程序的分支共有 5 条。处理完相应的程序后（SEG0 ~ SEG3，同时还包括 ERR），程序分支最后汇总到 COMM 标号位置。多分支跳转程序执行流程图如图 4-74 所示。

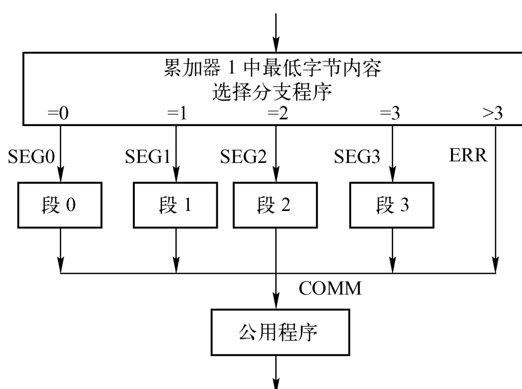



图 4-74　多分支跳转程序执行流程图

注意：多分支跳转语句在其后状态图编程分析中经常使用，所以需要很好地掌握。

### 3. 与 RLO 和 BR 有关的跳转指令

与 RLO 和 BR 有关的跳转指令有 JC、JCN、JCB、JNB。在执行 JCB、JNB 指令时，程序根据 RLO 的条件跳转至相应标号处，同时，将 RLO 置 1，并将 RLO 的内容复制到 BR 中。

JC 语句的具体使用（语句表程序）如下：

```
A   I 1.0
A   I 1.2
JC  JOVR          //如果 RLO =1,(I1.0 AND I1.2)则跳转至 JOVR 标号处
L   IW8           //如果 RLO =0,则执行装载和传输指令
```

```
          T    MW22         //将 IW8 的内容传输给 MW22
JOVR：A    I 2.1
```

说明：程序通过判断 RLO 之值，进行条件跳转，当 RLO 值为 1 时，程序跳转至标号为 JOVR 的程序段处。

### 4. 与信号状态位有关的跳转指令

这些跳转指令检查前一条指令的信号状态位：BR、OV、OS，满足条件的进行跳转。跳转指令有 JBI、JNBI、JO、JOS。

JO 语句的具体使用（语句表程序）如下：

```
          L    MW10
          L    3
          *I                // MW10 乘“3”
          JO   OVER         //如果值超出正确范围（OV＝1）,则跳转至 OVER 处
          T    MW10         //如果值没有超出正常范围,则执行以下语句
          A    M 4.0
          R    M 4.0        //将 M4.0 位清零
          JU   NEXT         //无条件跳转至 NEXT 处
OVER：AN   M 4.0
          S    M 4.0        //值超出范围后,置 M4.0 为 1
NEXT：NOP
```

说明：程序进行两个数的整数相乘，并判断相乘值是否溢出，溢出则置标志位 M4.0 为 1，否则，M4.0 清零。

### 5. 与条件码 CC0 和 CC1 有关的跳转指令

与条件码 CC0 和 CC1 有关的跳转指令实际上是观察运算结果与零值的关系，小于、等于或者大于零。

JP 语句的具体使用（语句表程序）如下：

```
          L    IW8
          L    MW12
          -I                //IW8 和 MW12 两数相减
          JP   POS.         //若结果为正,则跳转至 POS 处
          AN   M 4.0        //否则,执行以下程序
          S    M 4.0        //置位 M4.0
          JU   NEXT
POS：     AN   M 4.1
          S    M 4.1        //置位 M4.1
NEXT：NOP 0
```

说明：程序主要通过整数减法比较 IW8 和 MW12 中数值的大小，在 IW8 大于 MW12 时，通过 JP 指令，跳转至 POS 标号处，并置 M4.1 为 1；而在小于时，置 M4.0 为 1。其中，无条件跳转语句跳转至 NEXT 处，则是程序分支后的汇总处。

### 6. 梯形图和功能块图中的跳转指令

梯形图和功能块图中的跳转指令有无条件跳转和有条件跳转指令两种。其中条件跳转指

令的线圈必须受接点电路的控制，当它前面的逻辑操作结果 RLO =1 时，跳转指令执行。无条件跳转指令则无须受接点电路的控制。

**例 21** 试求和 $\sum_{k=1}^{20} k$ 。

求和程序如图 4-75 所示。

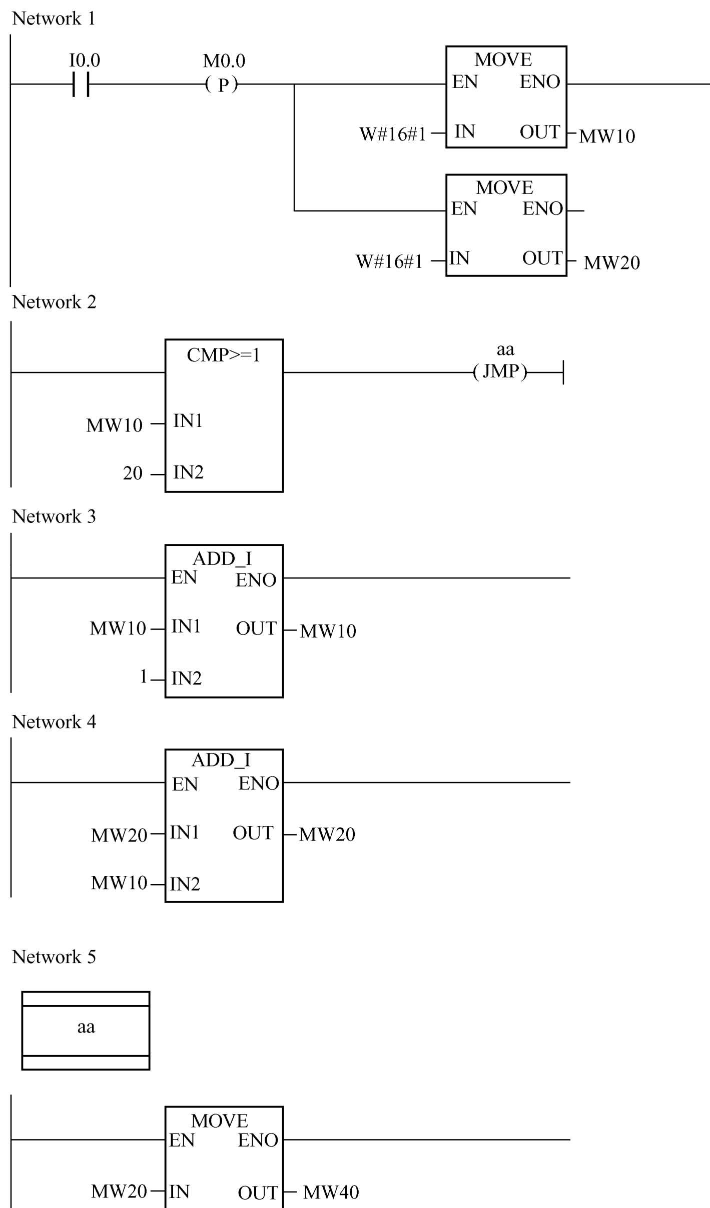



图 4-75　求和程序

说明：在程序的 Network1 中，首先对 MW10 和 MW20 赋初值 1，通过正边缘触发指令使 Network1 只执行一次，完成初值的设定；在 Network3 和 Network4 中通过整数加法指令，使 MW10 实

现从 1 到 20 的数值产生，而 MW20 则实现累加 20 个数的任务；通过 Network2 的条件跳转语句实现 20 以外数值的跳转；Network5 中，当数值大于等于 20 时，将累加结果送至 MW40 中。

**例 22** 试设计时钟脉冲发生器。要求输出位从 Q12.0 ~ Q13.7（16 位）分别输出频率范围从为 2 Hz ~ 0.000061 Hz 的脉冲。

时钟脉冲发生器的程序如图 4-76 所示。

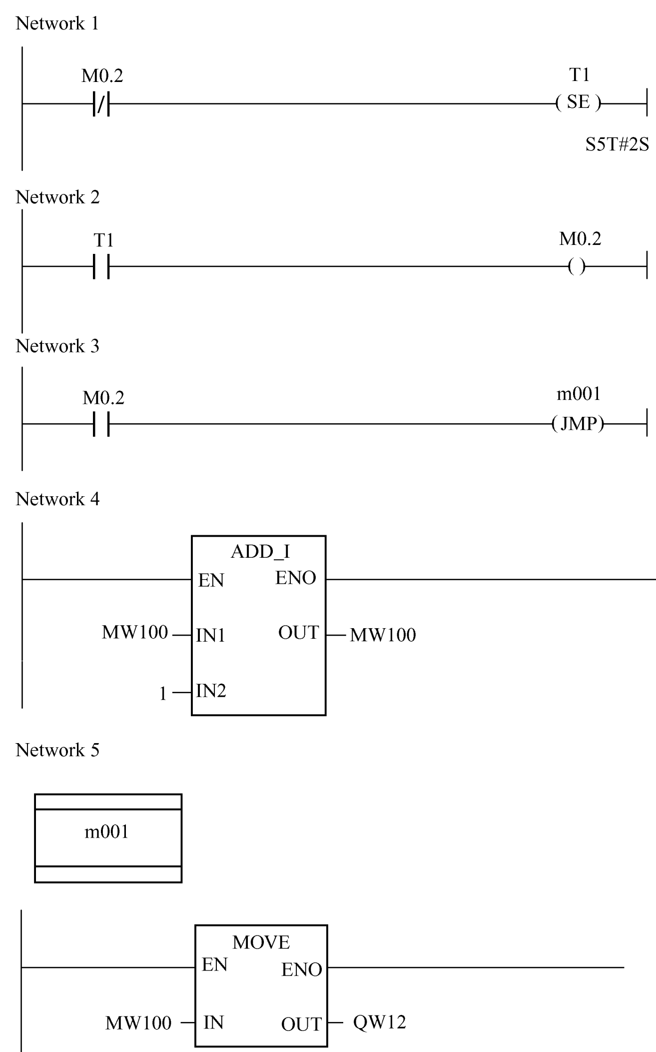



图 4-76 时钟脉冲发生器的程序

说明：程序中 Network1 和 Network2 表示扩展脉冲定时器 T1 每隔 250ms 产生一个负脉冲；Network4 只有当 T1 定时时间到的情况下，负脉冲产生时才执行，此时 RLO 之值为零，这样存储器 MW10 的内容加 1；Network5 表示存储器内容 MW100 传输到 QW12 中，那么因为 MW100 的不断累加，使输出位从 Q12.0 ~ Q13.7 分别输出频率范围从 2Hz ~ 0.000061Hz 的脉冲。

**7. 循环语句**

如果需要执行重复若干次的任务，则循环语句 LOOP 有这样的功能。在每执行一次 LOOP 指令时，累加器的值减 1；若减 1 后，累加器值非零，则跳转到 label 指定的标号处。

**例 23**　用循环指令求 5 的阶乘。

STL 程序如下：

```
      L     L#1          //将 32 位整数常数装入累加器 1 中,置阶乘的初值
      T     MD20         //保存入 MD20 中
      L     5            //循环次数装入累加器中
BB：  T     MW10         //循环次数保存于 MW10 中
      L     MD20
      *D                 // MD20 与 MW10 中内容相乘
      T     MD20         //乘积结果存入 MD20 中
      L     MW10
      LOOP  BB           //如果累加器中循环次数减 1 后大于零,则跳转至 BB 处
      ….                 //循环结束后,继续线性扫描
```

注意：循环次数不能为负数。LOOP 语句的流程图如图 4-77 所示。

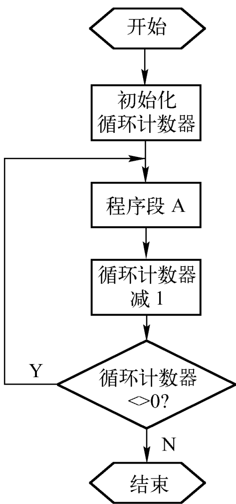



图 4-77　LOOP 语句的流程图

## 4.8.2　程序控制指令

程序控制指令是指逻辑调用指令和逻辑块结束指令。调用块和结束块同样可以是无条件的，也可以是有条件的。而逻辑块在 STEP 7 中实际为子程序，包括功能、功能块、系统功能和系统功能块。

程序控制指令如表 4-26 所示。

**表 4-26　程序控制指令**

| 梯　形　图 | 功 能 块 图 | 语句表指令 | 说　　明 |
|---|---|---|---|
| | | BE | 块结束 |
| | | BEU | 块无条件结束 |
| | | BEC | 块条件结束 |
| | | CALL FCn | 调用功能 |
| | | CALL FBn1，DBn2 | 调用功能块 |

（续）

| 梯 形 图 | 功 能 块 图 | 语句表指令 | 说 明 |
|---|---|---|---|
| | | CALL SFCn | 调用系统功能 |
| | | CALL SFBn1，DBn2 | 调用系统功能块 |
| —(CALL)—\| | CALL | CC FCn 或 SFCn | RLO = 1 时，条件调用 |
| —(CALL)—\| | CALL | UC FCn 或 SFCn | 无条件调用 |
| —( RET )—\| | REF | RET | 条件返回 |

在 CALL 指令中，FC、FB、SFC 和 SFB 是作为地址输入的，其地址可以是绝对地址，或者是符号地址。在调用 FB 和 SFB 时，必须提供与之相对应的背景数据块；而调用 FC 和 SFC 时，不需调用背景数据块。

在调用时，应将实参赋给被调用功能中的形参，并确保实参和形参数据类型相同。并且在 FC 和 SFC 的调用中，必须为所有形参指定实参，而调用 FB 和 SFB，则只需指定上次调用后必须改变的实参。

FB 功能块的具体调用（语句表程序）如下：

```
CALL    FB1,DB1           //调用 FB1,其背景数据块为 DB1
MAX:=MW10                 //MAX 为 FB1 定义的参数,将 MW10 之值赋值给 MAX
MIN:=MW20                 //将 MW20 之值赋值给 FB1 参数 MIN
POWER_ON:= I0.0           //将 I0.0 赋值给 FB1 参数 POWER_ON
POWER_OFF:=I0.1           //将 I0.1 赋值给 FB1 参数 POWER_OFF
```

说明：程序中调用了背景数据块 DB1，并将实参（“：=”之后的变量）赋给形参（“:”之前的变量）。功能块 FB 的具体用法见第 7 章。

梯形图无条件调用和条件调用的具体应用（见图 4-78a），功能块图的具体应用（见图 4-78b）。

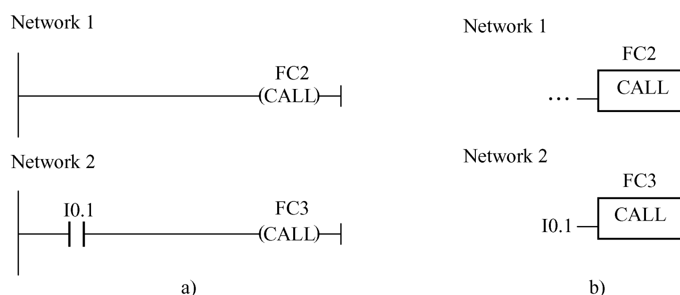



图 4-78　梯形图无条件调用和条件调用

实现上述相同功能的语句表程序如下：

```
UC    FC2          //无条件调用功能 FC2
A I   0.1
CC    FC3          //如果 RLO =1,条件调用功能 FC3
```

### 4.8.3 主控继电器指令

主控继电器是梯形图逻辑主控开关，用于控制信号流的关断。

MCRA 为激活 MCR 区指令，表明按 MCR 方式操作区域的开始，MCRD 为取消 MCR 区，表明按 MCR 方式操作区域的结束。且 MCRA 和 MCRD 要成对出现。MCRA 和 MCRD 指令之间的操作将根据 MCR 位的状态进行。若在其间有 BEU 指令，则 CPU 执行此指令，并结束 MCR 区域。若在其间有块调用指令，则激活状态不能继承到被调用的块中去，所以必须在被调用的块中重新激活 MCR 区域。

注意：为避免人员及财产损失，不能使用 MCR 指令代替硬件的机械主控继电器来实现紧急停车功能。

“MCR(” 为打开主控继电器区，在 MCR 堆栈中保存 RLO 值，“MCR)” 为关闭主控继电器区，在 MCR 堆栈中取出保存在其中的 RLO 值。“MCR(” 和 “MCR)” 指令必须成对出现。

MCR 指令可以嵌套使用，允许最大嵌套数为 8 级，因为 CPU 中有一个深度为 8 级的 MCR 嵌套。

主控继电器指令如表 4-27 所示。

表 4-27 主控继电器指令表

| 梯形图 | 功能图 | STL 指令 | 说明 |
| --- | --- | --- | --- |
| —(MCRA)—\| | MCRA | MCRA | 激活 MCR 区 |
| —(MCRD)—\| | MCRD | MCRD | 结束 MCR 区 |
| —( MCR<)—\| | —MCR< | MCR ( | 打开主控继电器区 |
| —( MCR>)—\| | —MCR> | MCR) | 关闭主控继电器区 |

主控指令在梯形图和功能块图中的具体应用如图 4-79a、b 所示。

实现上述相同功能的语句表程序如下：

```
MCRA         //激活 MCR 区
A  I 1.0
MCR(         //在 MCR 堆栈中保存 RLO,打开主控继电器区
A  I 4.0
 =  Q 8.0    //如果 MCR 位为 0,则不论 I 4.0 的状态如何,Q8.0 均被清零
L  MW20
T  QW10      //如果 MCR 位为 0,则 QW10 被清零
)MCR         //结束 MCR 控制区
MCRD         //关闭 MCR 区
A  I 1.1
 =  Q 8.1    //在 MCR 区之外的程序不受 MCR 位控制
```

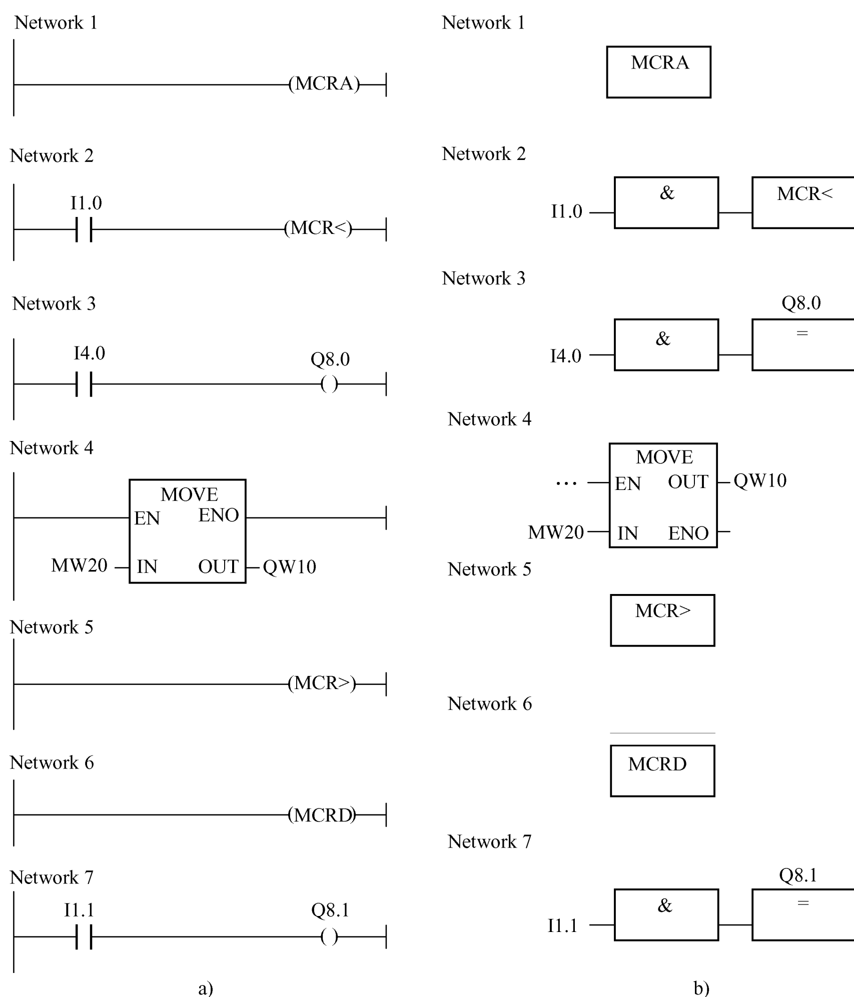



图 4-79　主控指令在梯形图和功能块图中的具体应用

## 习题

1. 要求利用移位指令使 8 盏灯以 0.2s 的速度自左向右亮起，到达最右侧后，再自右向左返回最左侧。如此反复。I0.0 =1 时移位开始，I0.0 =0 时移位停止。

2. 编写完成下面的算式：

$$\frac{50\times30-1}{50+1}$$

3. 易拉罐自动生产线上，需要统计出每小时生产的易拉罐数量。罐装易拉罐饮料一个接一个不断地经过计数装置。假设计数装置上有一个感应传感器，每当一听饮料经过时，就会产生一个脉冲。要求编制程序将 8h 的生产数量统计出来。

4. 在上题中，易拉罐的数量存储在存储器 MW0 ~ MW7 中，试编程找出其中最大的数。

5. 两台电动机功率分别为 1000W 和 2000W，开关 I0.0 控制电动机 1 的启停，开关 I0.1 控制电动机 2 的启停。试利用移位和加法指令实现运行电动机的功率显示。

## 4.9 应用实例

### 4.9.1 常用指令的综合用法

**例 24** 十字路口交通灯的设置如下：东西方向车流量较小，南北方向的车流量较大。因此南北方向的放行（绿灯亮）时间较长为 30s，东西方向的放行（绿灯亮）时间为 20s。当东西（或南北）方向的绿灯灭时，该方向的黄灯与另一方向的红灯一起以 1Hz 的频率闪烁 3s。以提醒司机和行人的注意。然后，立即开始另一个方向的放行。用一个控制开关（双向开关）对系统进行启停控制。交通灯工作时序图如图 4-80 所示。

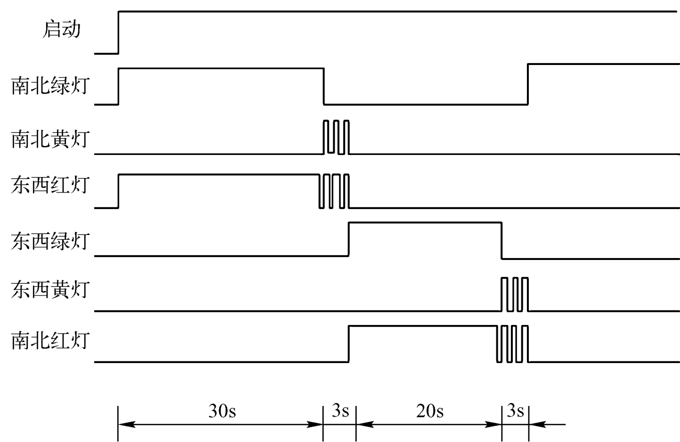



图 4-80 交通灯工作时序图

首先建立十字路口交通灯的变量表，如表 4-28 所示，在 STEP 7 中建立交通灯符号表，如图 4-81 所示。

**表 4-28 十字路口交通灯符号及变量表**

| PLC 地址 | WinCC 地址 | 符号表 | 数据类型 | 变量名 |
|---|---|---|---|---|
| I0. 0 | M10. 0 | start | BOOL | 启停开关 |
| Q4. 0 | Q4. 0 | SN_green | BOOL | 南北绿灯 |
| Q4. 1 | Q4. 1 | SN_yellow | BOOL | 南北黄灯 |
| Q4. 2 | Q4. 2 | SN_red | BOOL | 南北红灯 |
| Q4. 3 | Q4. 3 | EW_green | BOOL | 东西绿灯 |
| Q4. 4 | Q4. 4 | EW_yellow | BOOL | 东西黄灯 |
| Q4. 5 | Q4. 5 | EW_red | BOOL | 东西红灯 |

分析：从时序图中可以看出，需要使用四个定时器（T1 ~ T4），定时时间分别为 30s、3s、20s 和 3s。这里用脉冲定时器来实现定时器的循环起动（T1 起动 T2，T2 起动 T3，T3 起动 T4，T4 反过来起动 T1）。至于 1Hz 的闪烁频率则通过时间存储器 M0. 5 来实现（M0 设置为时钟存储字节）。时钟存储字节的第 5 位能产生频率为 1Hz 的脉冲。具体设定见第 8 章。

S7 Program(1) (Symbols) -- jiaotong...

| | Symbol | Address | Data type | Comment |
|---|---|---|---|---|
| 1 | Cycle Execution | OB 1 | OB 1 | |
| 2 | EW_green | Q 4.3 | BOOL | |
| 3 | EW_yellow | Q 4.4 | BOOL | |
| 4 | EW_red | Q 4.5 | BOOL | |
| 5 | SN_green | Q 4.0 | BOOL | |
| 6 | SN_yellow | Q 4.1 | BOOL | |
| 7 | SN_red | Q 4.2 | BOOL | |
| 8 | start | I 0.0 | BOOL | |

图 4-81　交通灯符号表

在 T1 时间段南北绿灯亮。在 T2 时间段南北黄灯亮，并且闪烁，所以可以通过 T2 和 M0.5 的串联实现。在 T3 时间段是南北红灯持续亮，在 T4 时间段南北红灯闪烁。南北红灯的控制为：首先将 T4 与 M0.5 的串联，再通过 T3 与之并联来实现。东西方向交通灯的编程思路同南北方向的交通灯。

交通灯梯形图程序如图 4-82 所示。

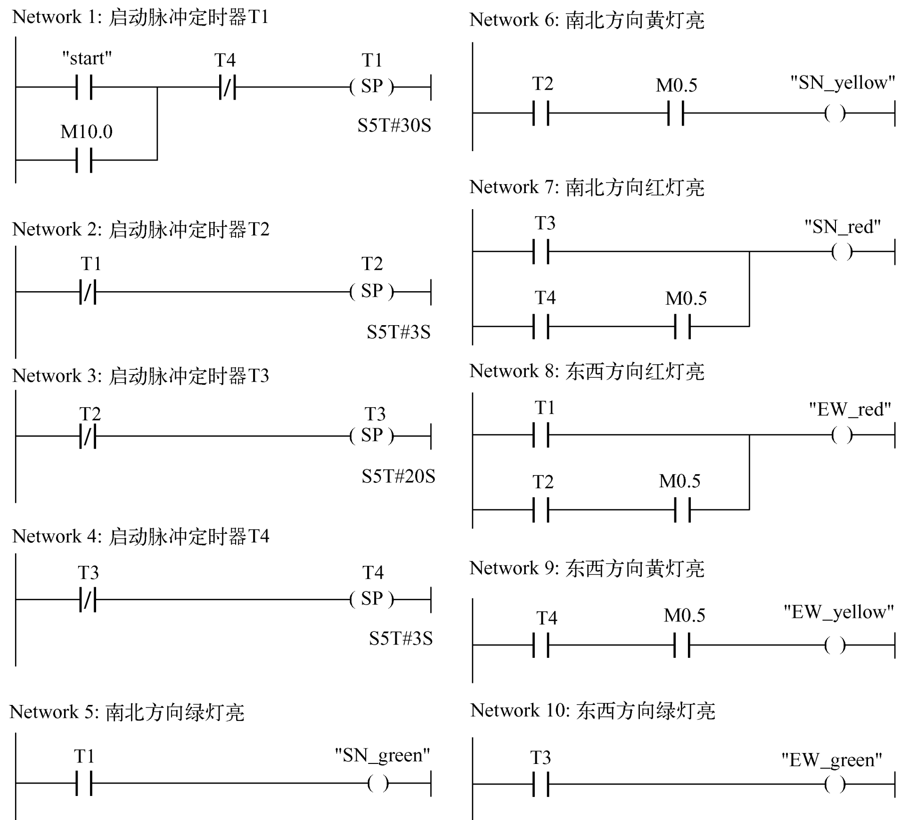



图 4-82　交通灯梯形图程序

交通灯控制的 WinCC 监控界面如图 4-83 所示。

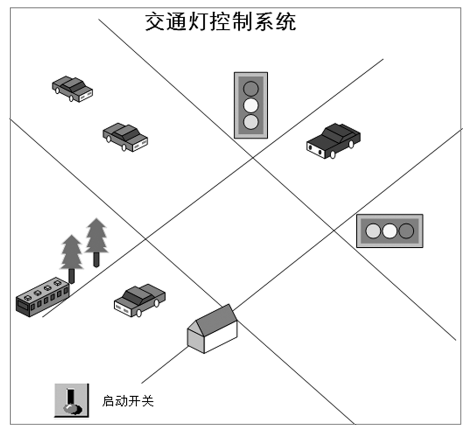



图 4-83　交通灯控制的 WinCC 监控界面

说明：程序 Network1、Network2、Network3 和 Network4 中的定时器构成振荡电路。在 Network1 中，上位机的控制开关与 PLC 的输入开关相并联，这样，无论是上位机还是 PLC 都可以控制系统的运行。由于上位机的输出地址与 PLC 的输出地址相同，所以在程序中，并不需要特殊处理。这样当启停开关拨到起动位置时，振荡电路开始工作；在 Network5 ~ Network10 中程序输出，控制各个方向的指示灯亮。

注意：

① 在仿真 PLC 调试时，当程序刚开始运行，会出现两个定时器同时起动的现象，此为程序 BUG，将所有定时器归零后，会正常运行。

② 程序也可以用接通延时定时器实现，略有不同而已。

**例 25**　走马灯的实现。要求：运用循环移位指令实现 8 个彩灯的循环左移和右移。其中 I0.0 为起停开关，MD20 为设定的初始值，MW12 为移位位数，输出为 Q0.0 ~ Q0.7。

首先建立循环彩灯变量表，如表 4-29 所示。在 STEP 7 中建立循环彩灯符号表，如图 4-84 所示。

**表 4-29　循环彩灯符号及变量表**

| PLC 地址 | WinCC 地址 | 符号表 | 数据类型 | 变量名 |
|---|---|---|---|---|
| I0.0 | M0.0 | start | BOOL | 启停开关 |
| I0.1 | M0.1 | L_rotate | BOOL | 左转 |
| I0.2 | M0.2 | R_rotate | BOOL | 右转 |
| MW12 | MW12 | interval | WORD | 移位位数 |
| MD20 | MD20 | I_value | DWORD | 初值 |

循环彩灯的梯形图程序如图 4-85 所示。

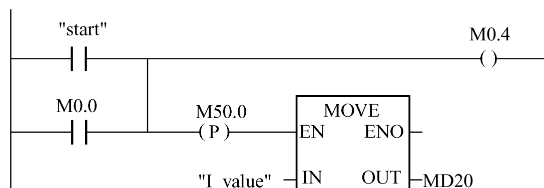
分析：首先建立定时振荡电路，振荡周期为 2.25 s，使得每次定时时间到后，循环移位指令开始移位。在循环移位指令的使用中运用了边缘触发指令，使循环移位在每个定时时间内只移位一次。在程序开始时，必须给循环存储器 MD20 赋初值，比如开始时，只有最低位

S7 Program(1) (Symbols) -- zzmadeng\SIM...

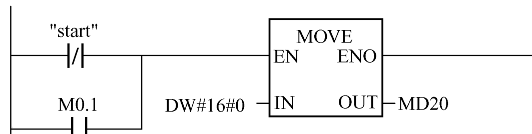
| | Status | Symbol | Address | Data type | Comment |
|---|---|---|---|---|---|
| 1 | | Cycle Execution | OB 1 | OB 1 | |
| 2 | | I_value | MD 20 | DWORD | 初值 |
| 3 | | interval | MW 12 | WORD | 移位位数 |
| 4 | | L_rotate | I 0.1 | BOOL | 左转 |
| 5 | | R_rotate | I 0.2 | BOOL | 右转 |
| 6 | | start | I 0.0 | BOOL | 起停开关 |
| 7 | | | | | |

图 4-84　循环彩灯符号表

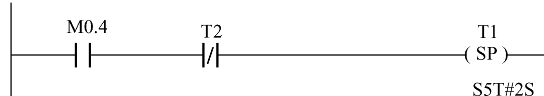
Network 1:启动并赋初值



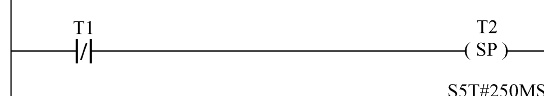

Network 2:清零



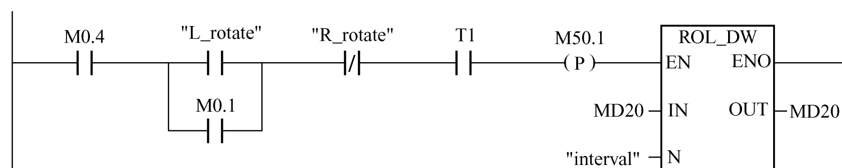

Network 3:启动定时器，构成振荡电路



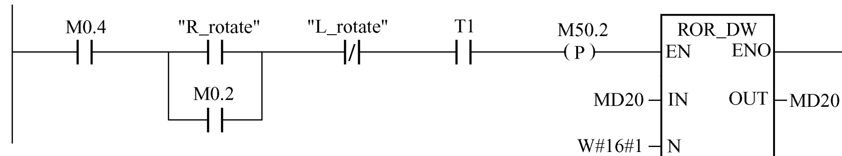

Network 4




Network 5:左转




Network 6:右转




Network 7:输出

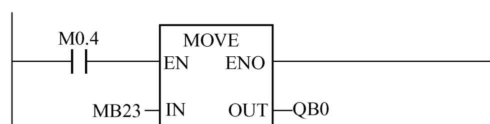



图 4-85　循环彩灯的梯形图程序

的彩灯亮（为 1），则初值设定必须为 DW#16#01010101（为了能循环显示，必须设定 MB20、MB21、MB22、MB23 中的值相同，均为 BW#16#01，否则，8 位彩灯轮流亮过后，彩灯会有段时间不亮）。

说明：在程序的 Network1 中，用起动开关通过边缘触发指令给 MD20 赋初值：I_ value；在 Network2 中，停止开关清除循环存储器 MD20 中的值；在 Network3 和 Network4 中，起动定时器，形成振荡电路；在 Network5 和 Network6 中，进行循环左移和右移。为移位正确，将边缘触发指令和循环移位指令配合使用，保证在每次定时器时间到时，只循环移位一次。注意：

① 为了连续移位，必须使循环移位指令的输入和输出相同。

② 通过这个程序的仿真 PLC 调试，可以很清楚地看到 MB20、MB21、MB22、MB23 中位的移动情况，深刻了解字节在双字中的排列情况：即低序号的排在高位，而高序号的排在低位。这点与人的常规印象相反。编程时需多加注意。

③ 赋初值语句需加边缘触发指令。

④ 由于循环移位指令的微分性质，必须和边缘触发指令联合运用。

本例中上位机和 PLC 都能对循环彩灯系统进行控制，所以只需将两者的输入信号相并联即可，因为可以在上位机中进行互锁编程，所以在 PLC 程序中，上位机输入控制并没有进行互锁。由于上位机中定义的中间存储位和 PLC 中的定义相同，所以在 PLC 程序中无需进行特殊处理。循环彩灯系统的 WinCC 监控界面如图 4-86 所示。

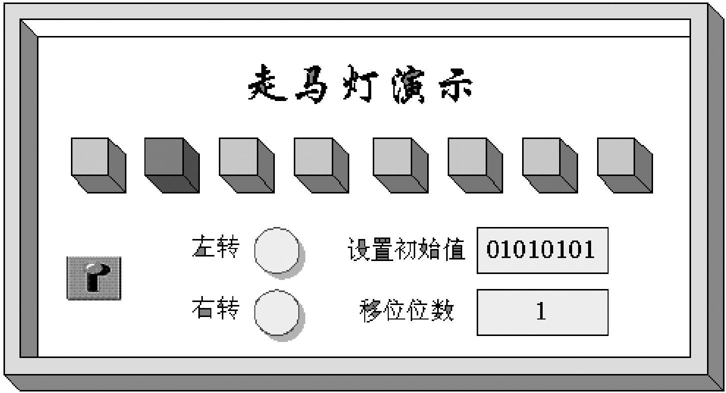



图 4-86　循环彩灯系统的 WinCC 监控界面

**例 26**　如图 4-87 所示为小车运输系统。有一部电动小车供 5 个加工点使用，对小车的控制要求如下：

1）起动按钮 I0.7 按下时，车停在某个加工点（工位：I0.0 ~ I0.4）。若没有用车呼叫（呼车：I1.0 ~ I1.4）时，工位允许呼叫指示灯亮（Q0.2），表示各工位可以呼车。

2）某工位呼车时，工位允许呼叫的指示灯灭，表示此后再呼车均无效。

3）停车位呼车则小车不动，当呼车位号大于停车位号时，小车自动向低位行驶（反转 Q0.1）；当呼车位号小于停车位号时，小车自动向高位行驶（正转 Q0.0）。当小车到达呼车位时自动停车。

4）小车到达呼车位时应停留 5s 供该工位使用，不应立即被其他工位呼走。

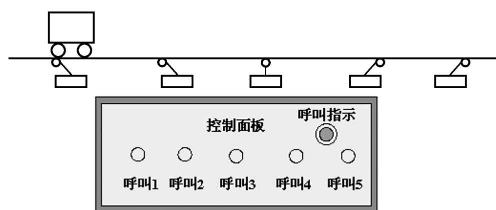


图 4-87　小车运输系统

试设计此系统。

首先根据系统要求画出控制流程图，如图 4-88 所示。

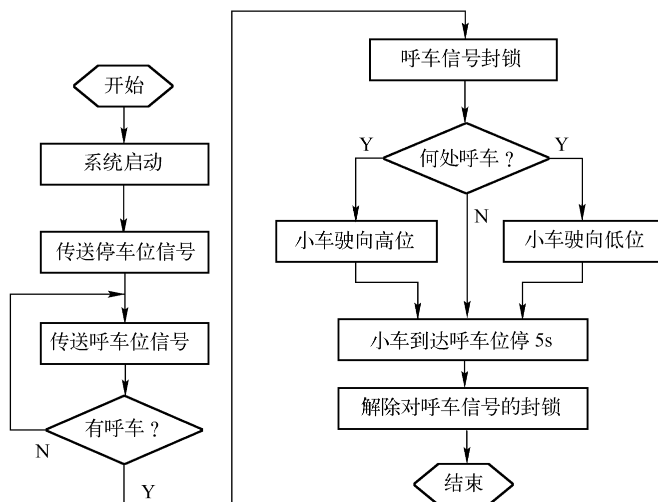


图 4-88　控制流程图

建立小车运输系统的变量表，如表 4-30 所示。

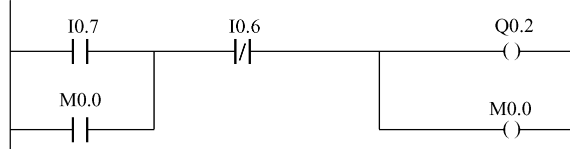
**表 4-30　小车运输系统的变量表**

| PLC 地址 | 数据类型 | 变量名 | PLC 地址 | 数据类型 | 变量名 |
|---|---|---|---|---|---|
| I0.7 | BOOL | 起动按钮 | I1.1 | BOOL | 呼车位 2 |
| I0.6 | BOOL | 停止按钮 | I1.2 | BOOL | 呼车位 3 |
| I0.0 | BOOL | 工位 1 | I1.3 | BOOL | 呼车位 4 |
| I0.1 | BOOL | 工位 2 | I1.4 | BOOL | 呼车位 5 |
| I0.2 | BOOL | 工位 3 | Q0.0 | BOOL | 正转 |
| I0.3 | BOOL | 工位 4 | Q0.1 | BOOL | 反转 |
| I0.4 | BOOL | 工位 5 | Q0.2 | BOOL | 允许呼车指示灯 |
| I1.0 | BOOL | 呼车位 1 | | | |

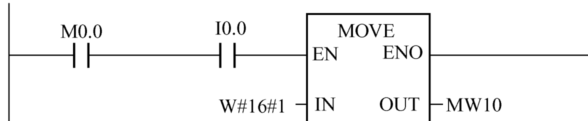
运输小车梯形图程序如图 4-89 所示。

分析：在设计中，首先将小车所在的工位号传送给存储器 MW10，再将呼车的工位号传送给存储器 MW12，两者相比较，当呼车的位号小于停车位号时，小车正转，反之，小车反

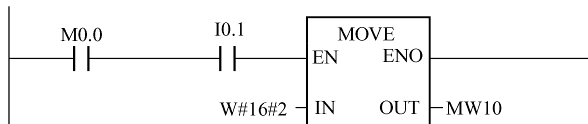
Network 1: 启动，指示灯亮



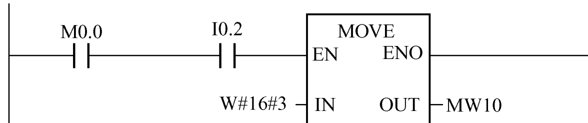

Network 2: 传输工位1



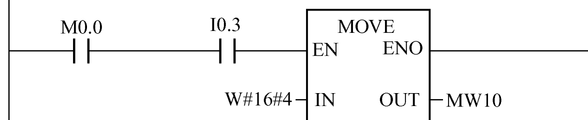

Network 3: 传输工位2



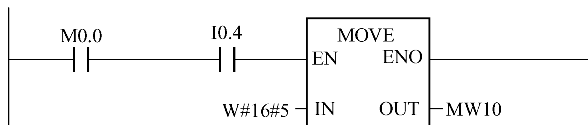

Network 4: 传输工位3



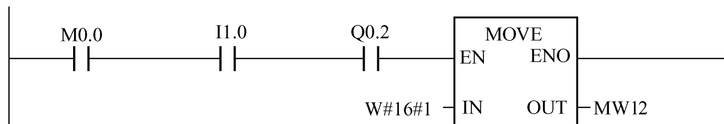

Network 5: 传输工位4



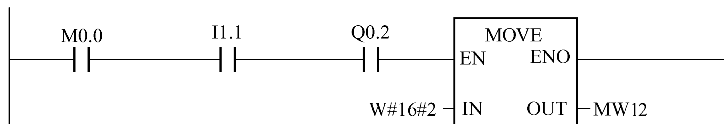

Network 6: 传输工位5




Network 7: 传输呼车位1




Network 8: 传输呼车位2




Network 9: 传输呼车位3

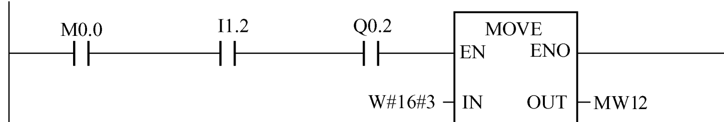
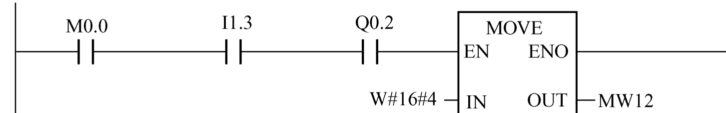
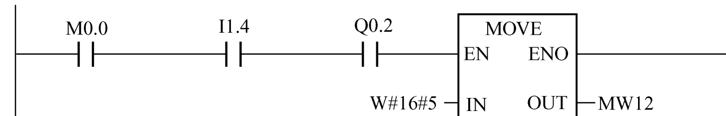



图 4-89　运输小车梯形图程序

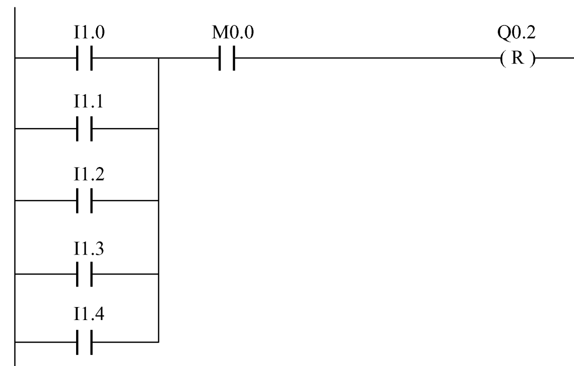
Network 10: 传输呼车位4



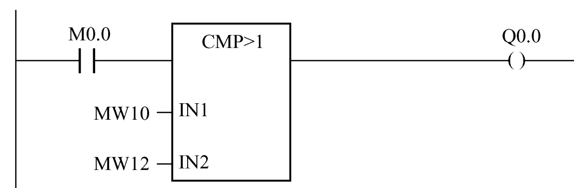

Network 11: 传输呼车位5



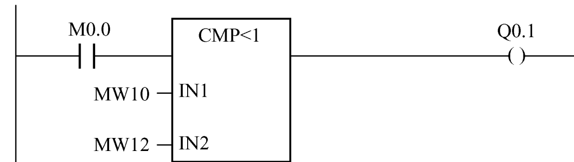

Network 12: 呼叫并使指示灯灭



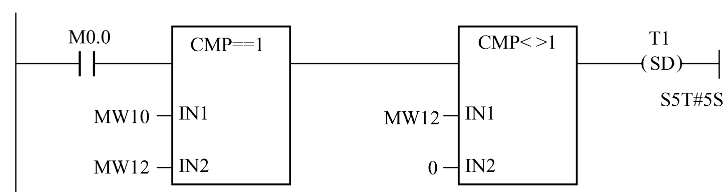

Network 13: 正转




Network 14: 反转




Network 15: 停车并延时




Network 16: 使用完毕信号灯亮

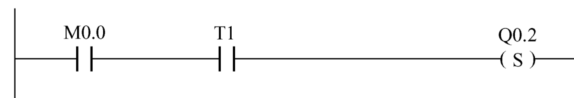



图 4-89　运输小车梯形图程序（续）

转。若呼车位号等于停车位，则起动定时器 T1 延时 5s，延时时间到，呼车信号允许指示灯亮，并取消对呼车信号的封锁。

程序中要注意，在允许呼车的前提条件下，若有呼叫信号，则将指示灯点亮，封锁其他呼叫信号。而传递呼车信号必须在允许呼车指示灯（Q0.2 = 1）的条件下，才能传递给 MW12。

说明：程序的 Network1 中，按钮 I0.7 使输出 M0.0 自锁，并使允许呼车指示灯亮；在 Network2 ~ Network6 中，根据位置传感器 I0.0 ~ I0.4，传递小车的停车位至 MW10 中；在 Network7 ~ Network11 中允许呼车的情况下，将呼车位置传递给 MW12 中；Network12 表示呼车之后，呼车允许指示灯灭；Network13 和 Network14 中，通过比较 MW12 和 MW10 的值，使小车正转或反转；Network15 表示小车到达呼车位置后，停止等待 5s，供工位使用，另一方面也避免了启动后执行程序时 MW10 和 MW12 均等于零导致的错误；Network16 中，等待 5s 过后，重新点亮呼车允许指示灯。

注意：比较指令与数值比较时，为整数值。而移位指令的数值形式可以用各种进制。

### 4.9.2 ET200M 的使用

**例 27** 智能家居分布式灯控系统。该系统利用 ET200M 通过 SIMIT 仿真软件模拟了智能家居中分布式灯控系统，即通过操作分布在不同地点的按钮（如书房、卧室、厨房），一起控制安装在大厅中的灯具。其 SIMIT 仿真效果图如图 4-90 所示。




图 4-90　SIMIT 仿真效果图

设计步骤：

1）先进行硬件组态。打开 STEP 7，新建一个工程，插入一个 300 站。插入 300CPU 模块时，需要选择新加入一个 PROFIBUS 网络，默认选择 DP 的地址为 2。在硬件目录 PROFIBUS－DP 选项中选择接口模块 IM153－2，添加到 PROFIBUS 网络上，加入 ET200M 时按照图 4-91 所示的标注位置选择相应的模块。

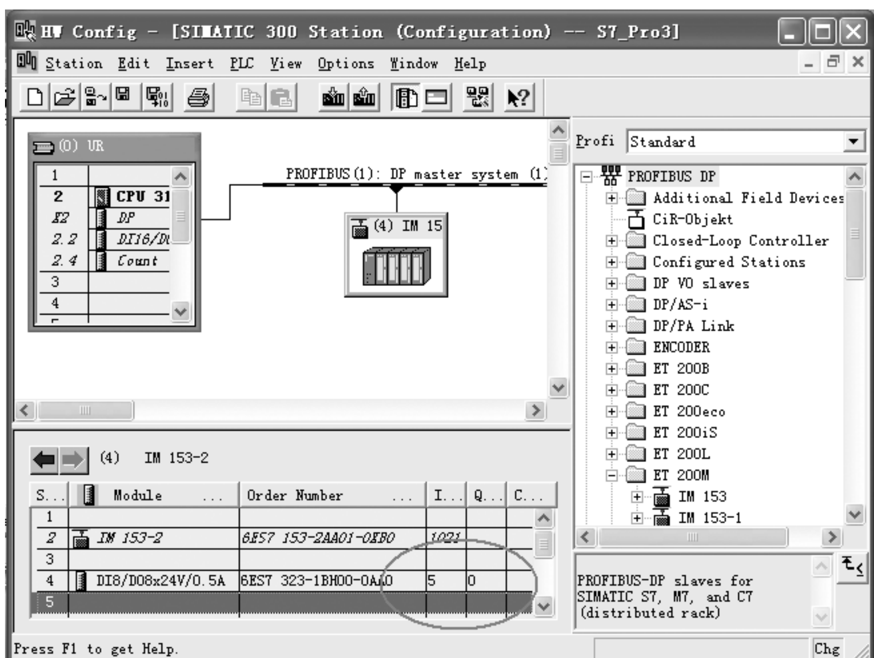



图 4-91　硬件组态图

2）在 ET200M 组态时，需要注意的是：DP 地址必须选择一个与主站不同的地址。ET200M 的地址设置为：在拨码设置上，拨在右面的开关数值叠加即为其地址值；在插入 IO 扩展模块时需要注意其 I/O 地址的选择，然后在程序中根据相应的地址进行编程。

3）SIMIT 仿真对象程序设计过程中的若干注意事项如下：

① 前台和后台程序设计过程中对象命名的大小写要一致，如图 4-92 所示为 SIMIT 后台对象的命名，如图 4-93 所示为 SIMIT 前台对象的命名。

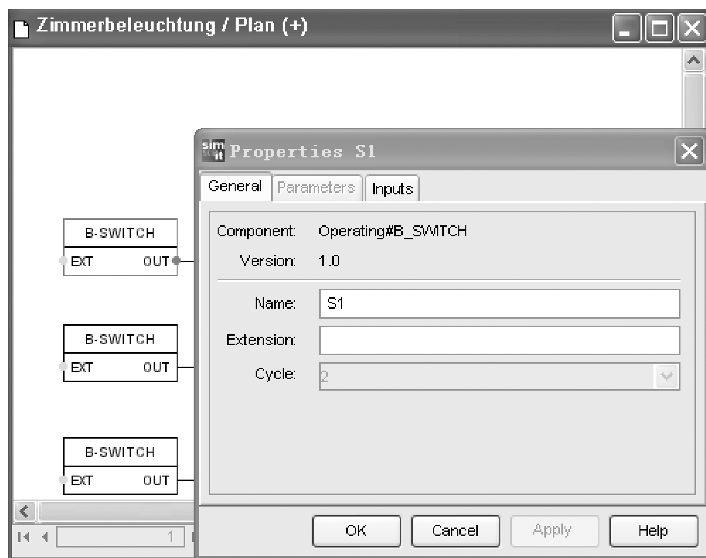



图 4-92　SIMIT 后台对象的命名

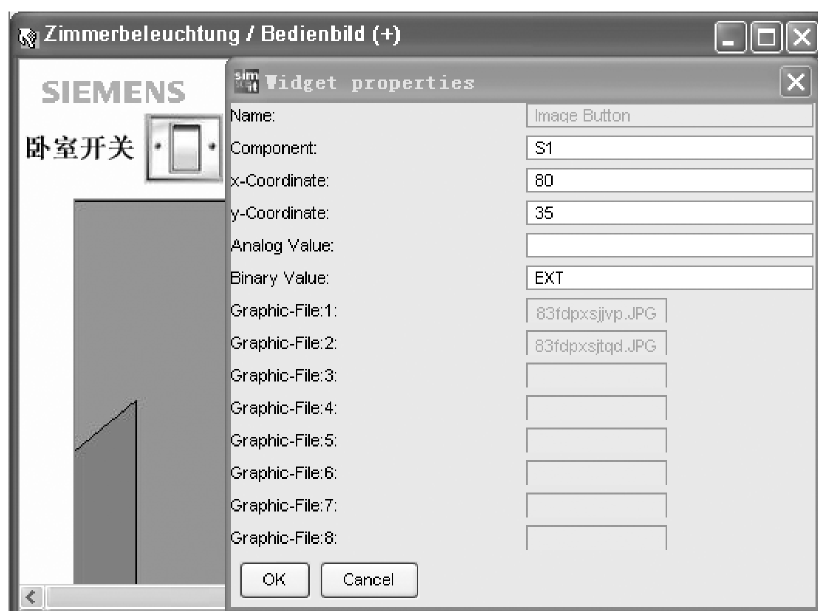



图 4-93　SIMIT 前台对象的命名

② SIMIT 仿真对象接口部分 MPI 设计的参数设置如图 4-94 所示。

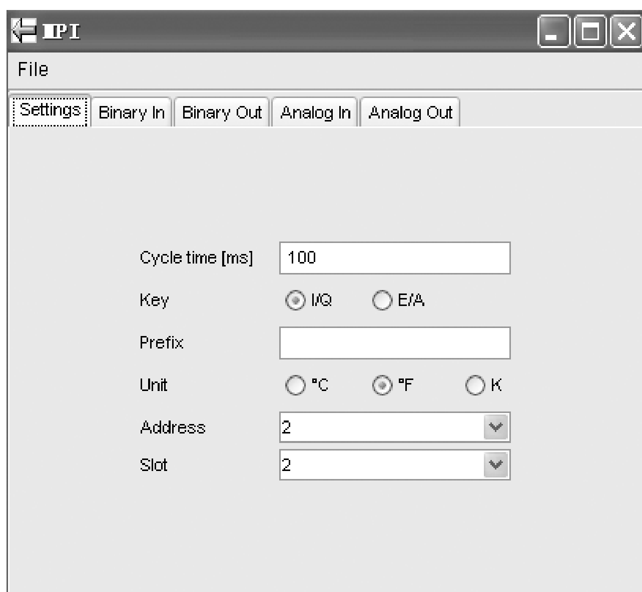



图 4-94　SIMIT 仿真对象接口部分 MPI 设计的参数设置

③ 在 SIMIT 中使用的输入资源 I0.0、I0.1、I0.2 不能和在 S7 中实际分配的输入资源 I5.0～I5.7 相冲突（ET200M 的地址）。SIMIT 中的输入地址分配如图 4-95 所示。这一点非常重要，如果真实输入地址和仿真输入地址一样，仿真输入将不起作用。

4）程序设计。设计方法有三种：

方法 1：用开关逻辑实现灯控系统。控制的逻辑真值表如表 4-31 所示。

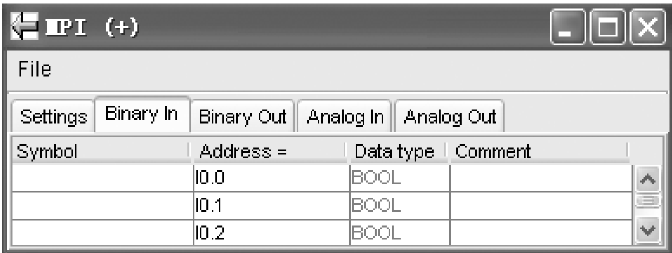



图 4-95　SIMIT 中的输入地址

**表 4-31　逻辑真值表**

| S0 | S1 | S2 | Y0 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 |

根据真值表可以列出逻辑函数关系式：

$$Y0=\overline{X0}\cdot\overline{X1}\cdot X2+\overline{X0}\cdot X1\cdot\overline{X2}+X0\cdot\overline{X1}\cdot\overline{X2}+X0\cdot X1\cdot X2$$

智能灯控系统程序 1 如图 4-96 所示。

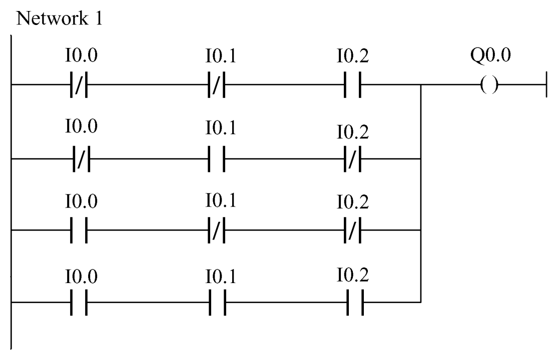



图 4-96　智能灯控系统程序 1

程序相当简单，ET200M 的三个输入开关分别连接在书房、卧室和厨房中。CPU 对其进行控制。开关处于乒乓状态，灯如果处于开的状态，则在任意一处位置按动开关，即可以使灯关，反之亦然。

方法 2：用边沿触发指令的上升沿和下降沿，同样可以进行灯控系统的设计。特点是不论开关是接通或者断开，即将输出取反。智能灯控系统程序 2 如图 4-97 所示。

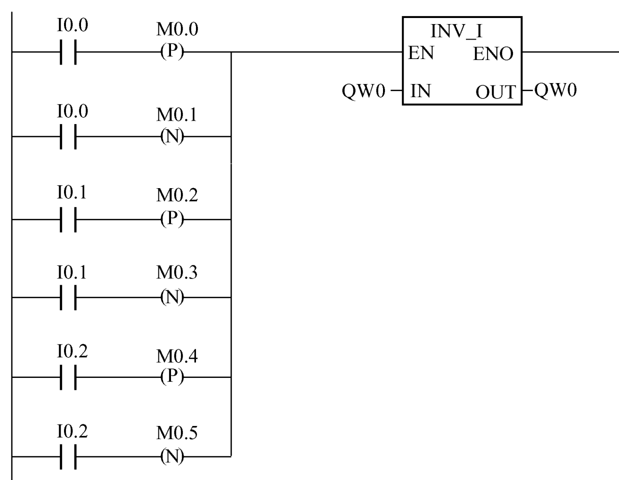



图 4-97　智能灯控系统程序 2

方法 3：使用比较指令，当存储器 IW0 中内容与 MW10 中内容不同时，将输出取反，同时将新内容 IW0 传送给 MW10。智能灯控系统程序 3 如图 4-98 所示。

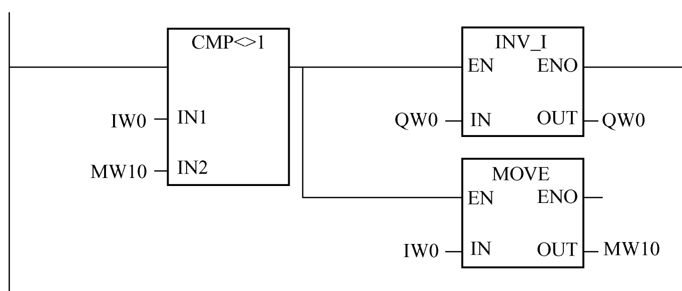



图 4-98　智能灯控系统程序 3

### 4.9.3　变频器的使用

**1. 变频调速系统的基本功能（数字量输入控制）**

变频调速系统的基本功能是实现对频率、转速等参数的自动测量与控制。

1）系统硬件配置如下：系统选用了西门子 S7 300 系列 PLC（CPU313C－2DP），它具有 PROFIBUS－DP 网络通信端口和 MPI 接口，可将其作为系统主站；控制系统的其他功能模块分别为：电源模块 PS307（2A）、数字量输入模块 SM322、模拟量输入/输出模块 SM334。变频器控制电动机的硬件组态如图 4-99 所示。

2）PLC 与变频器的接线如下：变频器由交流 220V 控制，其数字输入端 5、6、7 分别与 PLC 的数字量输出端相连（地址为 QW124、QW125），通过数字量输出模块对电动机进行正反转控制；仿真模块用于模拟控制开关，包括通电、正转和反转开关；变频器的模拟量输入端 3、4 接 PLC 的模拟量输入/输出模块 SM334 的输出端（地址为 PQW288），通过这个模块设置电动机转速。变频器的模拟量输出端 12、13 同样接 PLC 的模拟量输入/输出模块 SM334 的输入端（地址为 PIW288），通过这个模块取得电动机的转速；数字量输入模块 SM322（电源为交流 220V）与交流接触器线圈相连。控制变频器的通电。

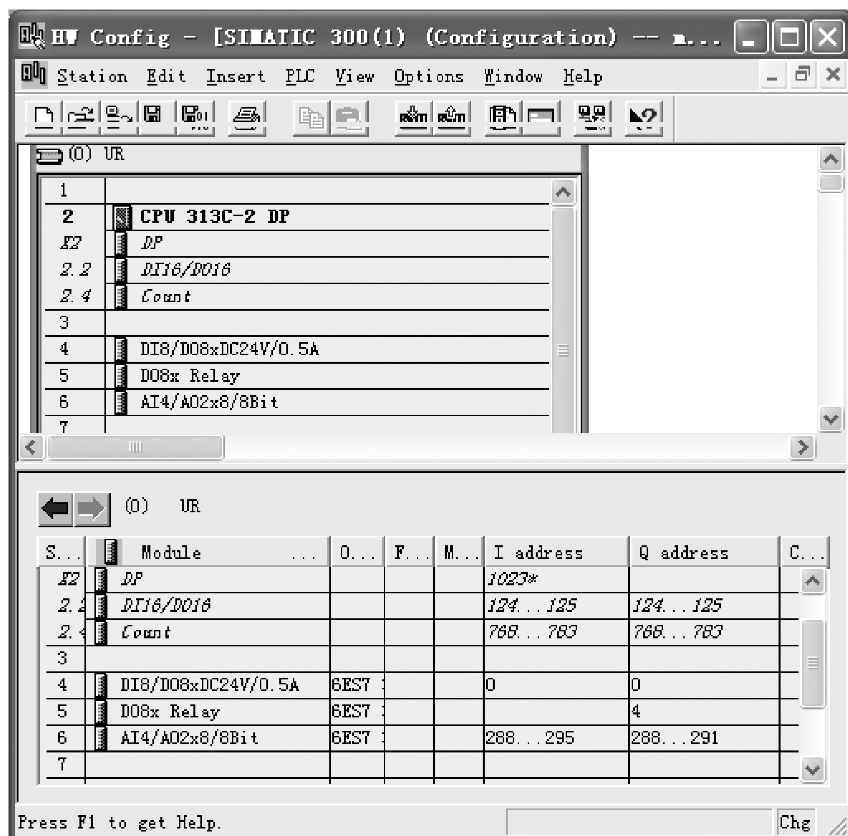



图 4-99　变频器控制电动机的硬件组态

3）变频器的参数设定。注意：在变频器设置中要先进行电动机和变频器的基本设定。除了基本设定外，变频器的其他设定为：P0700 = 2（选择命令源：端子排输入），P0701 = 1（数字输入 1 功能：1——正转或停止），P0702 = 2（数字输入 2 功能：2——反转），P0703 = 9（数字输入 3 功能：9——故障复位），P1000 = 2（模拟设定值）。

4）建立变频调速系统的变量表，如表 4-32 所示。

**表 4-32　变频调速系统的变量表**

| PLC 地址 | WinCC 地址 | 变量名 | PLC 地址 | WinCC 地址 | 变量名 |
|---|---|---|---|---|---|
| I0.0 | M0.0 | 通电开关 | Q124.0 | Q124.0 | 正转线圈 |
| I0.1 | M0.1 | 正转/停止开关 | Q124.4 | Q124.4 | 反转线圈 |
| I0.2 | M0.2 | 反转/停止开关 | PIW288 | MW8 | 取得转速 |
| Q4.4 | Q4.4 | 通电接触器线圈 | PQW288 | MW10 | 设置转速 |

5）编程。如图 4-100 所示为变频调速基本功能的梯形图。

说明：通过 PLC 输入或 WinCC 界面中的控制按钮使电动机正转、反转或停止（对应控制开关并联）。程序 Network4 中电动机转速可以通过 PQW288 设定。Network5 中通过 PIW288 取出电动机转速。

6）在 WinCC 中可以监控频率（转速）曲线。系统还采用 MPI 协议与 PLC 通信，通过 WinCC 对系统运行进行监控。变频器基本功能监控界面如图 4-101 所示。

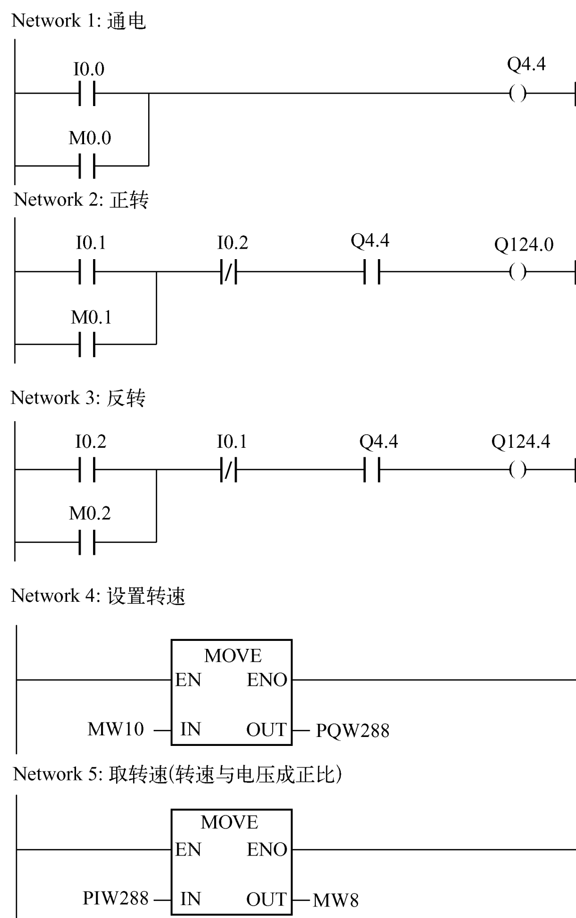



图 4-100　变频调速基本功能的梯形图

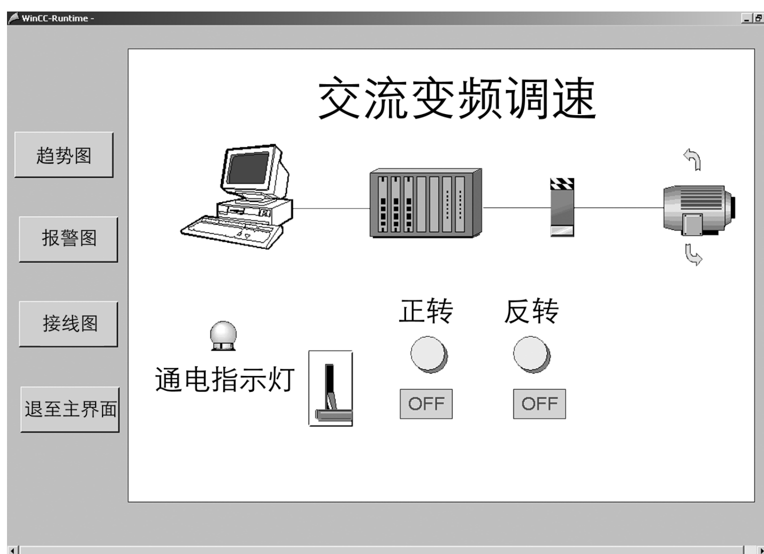



图 4-101　变频器基本功能监控界面

### 2. 多级频率调速

工业生产中经常利用固定频率进行变频调速。如工业洗涤机的脱水过程就需要设定洗涤曲线。机件加工也需要加工曲线的设定，因而需要利用固定频率（多级）调速。

PLC 与变频器的接线同前。首先设置多级频率调速系统的变量表，如表 4-33 所示。

**表 4-33　多级频率调速系统的变量表**

| PLC 地址 | WinCC 地址 | 变量名 | PLC 地址 | WinCC 地址 | 变量名 |
|---|---|---|---|---|---|
| I0. 0 | M10. 0 | 通电开关 | Q124. 0 | Q124. 0 | 固定频率 1 输出 |
| I0. 1 | M10. 1 | 固定频率 1 | Q124. 4 | Q124. 4 | 固定频率 2 输出 |
| I0. 2 | M10. 2 | 固定频率 2 | Q124. 2 | Q124. 2 | 固定频率 3 输出 |
| I0. 3 | M10. 3 | 固定频率 3 | PIW288 | MW0 | 取得转速 |
| Q4. 4 | Q4. 4 | 通电接触器线圈 | | | |

变频器的参数设定如下：P0700 = 2（模拟数字输入）；P1000 = 2（模拟设定值）；P0701，P0702，P0703 = 16（固定频率设定值：直接选择 + ON 命令）；P1001 = 20，P1002 = 15，P1003 = 10（固定频率 1、2、3，同时选择可将固定频率叠加）；P1016，P1017，P1018 = 2（直接选择 + ON 命令）；P1020 = 722. 0，P1021 = 722. 1，P1022 = 722. 2（固定频率选择来源：数字输入 1、2、3）；P1130 = 5，P1120 = 10，P1131 = 5（S 型曲线上升定义）。

多级频率调速的梯形图如图 4-102 所示。多级频率调速 WinCC 控制界面如图 4-103 所示。

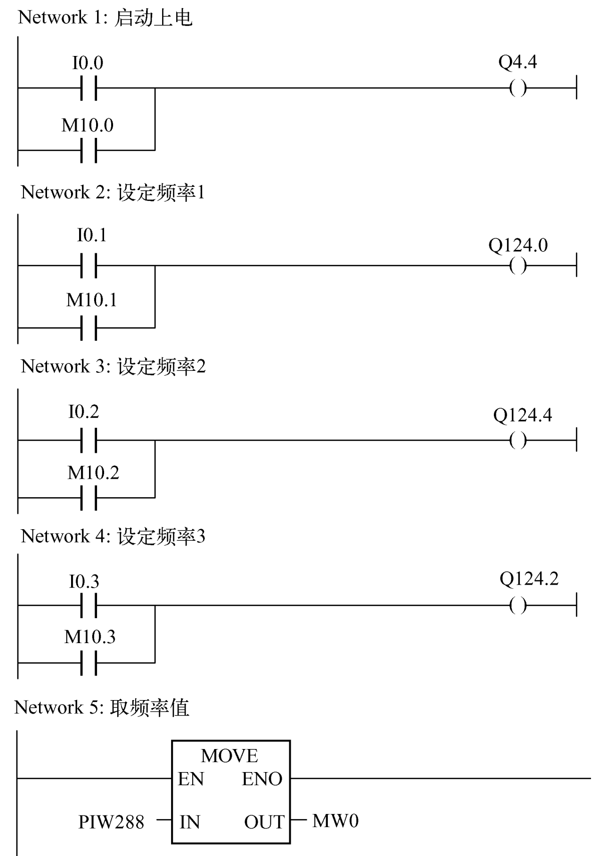



图 4-102　多级频率调速的梯形图

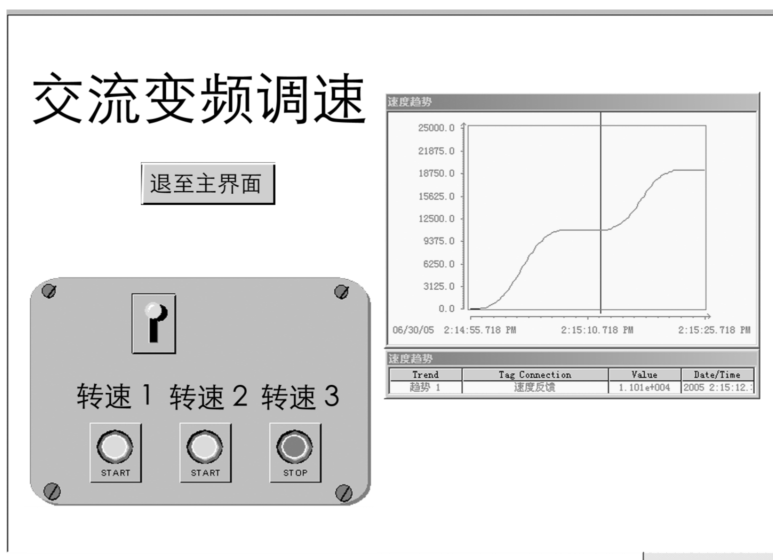



图 4-103　多级频率调速 WinCC 控制界面

3. 通过 PROFIBUS_DP 通信控制的交流变频调速

PROFIBUS_DP 是一种高速、经济的设备级网络，主要用于现场控制器与分散 I/O 信号之间的通信，可以满足交直流调速系统快速相应的时间要求。由于其可靠性高、实时性好，已被几乎所有的生产厂商和用户所接受。

1）硬件组态。在交流调速系统中，因为主站是通过 PROFIBUS_DP 连接从站变频器 MM420，所以必须有 PROFIBUS_DP 接口模块安装在变频器上。硬件组态如图 4-98 所示。

组态步骤如下：

- 插入 300 站，依次插入电源、CPU。
- 在插入 CPU 时，会弹出 PROFIBUS 组态界面，新建 PROFIBUS（1），并设置 PROFIBUS 站地址为 2，“Operating Mode”设置为 DP master。
- 组态从站时，将 PROFIBUS_DP 中 SIMOVERT 文件夹下的 MICROMASTER 4 挂到 PROFIBUS 总线上，并设置其站地址为 4。
- 点击变频器图标，在下栏中插入 PPO 3，如图 4-104 所示。

注意：MICROMASTER 4 仅支持 PPO 1 和 PPO 3。

2）MM420 控制字和状态字的含义。MM420 中的通信区为 PKW 和 PZD。其中设定值、控制字以及反馈数据存放在 PZD 区，PKW 数据区是变频器运行时要定义的一些功能码，如最大频率、基本频率、加/减速时间等。PKW 数据区传递必须用打包和解包程序，而 PZD 区的数据传递可以直接传递。

PZD 第一个控制字 SKW 设定含义如表 4-34 所示。

对于变频器收到的控制字，其位 10 必须设置为 1。如果位 10 是 0，控制字设置不起作用。

PZD 任务报文的第 2 个字是主频率设定值（HSW）。

PZD 应答报文的第 1 个字是变频器的状态字（ZSW）。PZD 状态字（ZSW）的含义如表 4-35 所示。

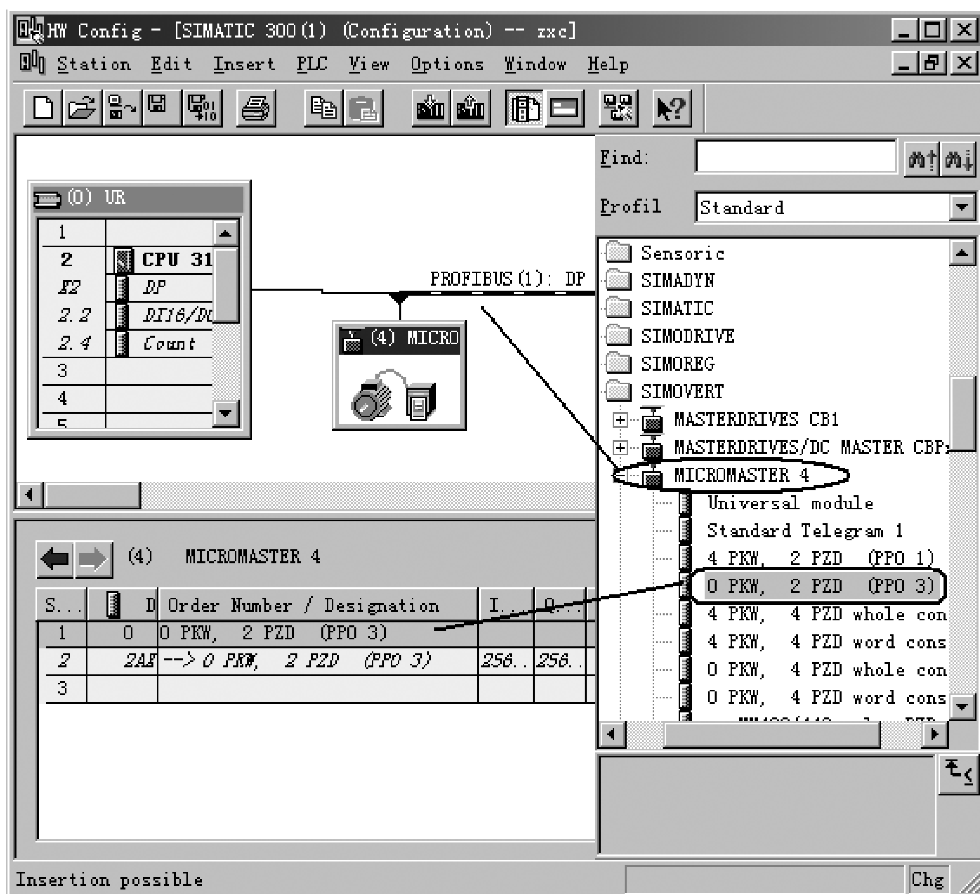



图 4-104　硬件组态图

表 4-34　PZD 控制字（SKW）

| | | | |
|---|---|---|---|
| 位 00 | “On（斜坡上升）/OFF1（斜坡下降）” | 0 否 | 1 是 |
| 位 01 | “OFF2：按惯性自由停车” | 0 是 | 1 否 |
| 位 02 | “OFF3：快速停车” | 0 是 | 1 否 |
| 位 03 | “脉冲使能” | 0 是 | 1 否 |
| 位 04 | “斜坡函数发生器（RFG）使能” | 0 是 | 1 否 |
| 位 05 | “RFG 开始” | 0 是 | 1 否 |
| 位 06 | “设定值使能” | 0 是 | 1 否 |
| 位 07 | “故障确认” | 0 是 | 1 否 |
| 位 08 | “正向点动” | 0 是 | 1 否 |
| 位 09 | “反向点动” | 0 是 | 1 否 |
| 位 10 | “由 PLC 进行控制” | 0 是 | 1 否 |
| 位 11 | “设定值反向” | 0 是 | 1 否 |
| 位 12 | 未使用 | | |
| 位 13 | “用电动电位计（MOP）升速” | 0 是 | 1 否 |
| 位 14 | “用 MOP 降速” | 0 是 | 1 否 |
| 位 15 | 本机/远程控制 | | |

表 4-35　PZD 状态字（ZSW）

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 位 0 | 变频器准备 | 0 | 否 | 位 7 | 变频器报警 | 0 | 否 |
| 位 1 | 变频器运行准备就绪 | 0 | 否 | 位 8 | 设定值/实际值偏差过大 | 0 | 是 |
| 位 2 | 变频器正在运行 | 0 | 否 | 位 9 | PZDl（过程数据）控制 | 0 | 否 |
| 位 3 | 变频器故障 | 0 | 是 | 位 10 | 已达到最大频率 | 0 | 否 |
| 位 4 | OFF2 命令激活 | 0 | 是 | 位 11 | 电动机电流极限报警 | 0 | 是 |
| 位 5 | OFF3 命令激活 | 0 | 否 | 位 12 | 电动机抱闸制动投入 | 0 | 是 |
| 位 6 | 禁止 on（接通）命令 | 0 | 否 | 位 13 | 电动机过载 | 0 | 是 |

PZD 应答报文的第 2 个字是主要的运行频率实际值（HIW）。

3）建立通过 PROFIBUS 控制的交流调速系统的变量表，如表 4-36 所示。

表 4-36　通过 PROFIBUS 控制的交流调速系统的变量表

| PLC 地址 | WinCC 地址 | 变量名 |
|---|---|---|
| MW40 | MW40 | 控制字设定 |
| MW42 | MW42 | 电动机频率设定 |
| MW20 | MW20 | 状态字 |
| MW22 | MW22 | 实际频率值 |

4）变频器的设置为：P700 = 6（命令源：远程控制从 CB 来），P918 = 4（站号设定），P100 = 6（频率设定源：从 CB 来）。

5）编程。通过 PROFIBUS 控制变频器的梯形图，如图 4-105 所示。

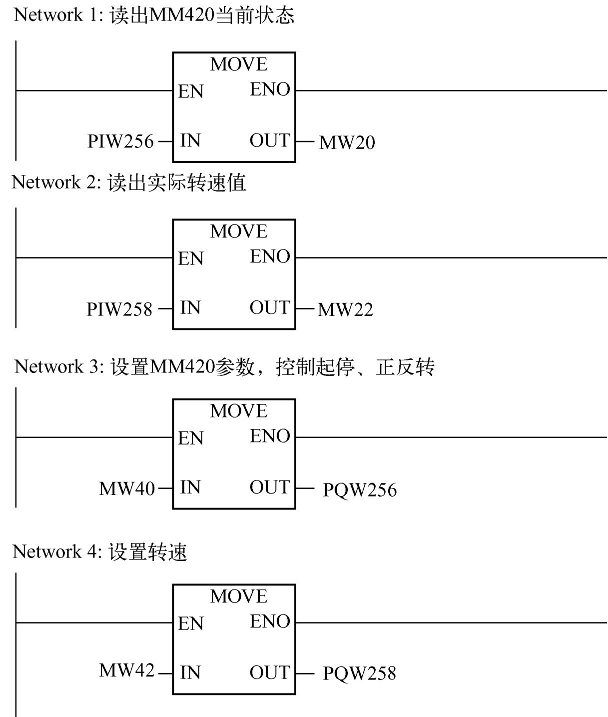



图 4-105　通过 PROFIBUS 控制变频器的梯形图

6）通过 PROFIBUS 控制变频器的控制界面，如图 4-106 所示。



图 4-106　通过 PROFIBUS 控制变频器的控制界面

# 第 5 章　调 试 方 法

STEP 7 提供可视化的在线调试功能。在 STEP 7 中完成的硬件组态和用户程序必须通过电缆下载到 PLC 中，经过软硬件的联调成功后，才能最终完成控制任务。调试内容及其工具如表 5-1 所示。根据工具的不同，我们将分别介绍其调试的方法。运用恰当的调试方法会缩短项目的调试周期。此外还介绍了结构化程序的调试方法和仿真软件 S7 - PLCSIM 的应用。

表 5-1　调试内容及其工具

| 调 试 内 容 | 工　　具 |
|---|---|
| 可视化的硬件检查 | CPU 模块上的 LED 指示灯 |
| 调试硬件 | 硬件组态窗口 |
| 调试软件 | 在程序状态中修改变量 |
| | 在变量表中修改变量 |
| | 诊断缓冲区 |
| | 强制变量 |
| | 参考表 |

## 5.1　利用 LED 指示灯调试

几乎在每块硬件模块的面板上都有指示灯，它直观地显示出模块当前的状态及其他基本的诊断信息。用户可以观察模块的指示灯并结合“Monitor/Modify Variables”工具来调试硬件。在调试过程中，需熟悉 CPU 上指示灯的指示信息，主要是以下 7 个指示灯：

① SF（红色）：程序错误时亮，或带有诊断功能的模块错误时亮。例如，算术运算或定时器出错，外部输入/输出的故障或错误等。

② BF（红色）：网络组态时总线错误指示灯亮，集成有 DP 接口的 CPU 才会有这个 LED 灯。

③ DC 5V（绿色）：指示 5V 直流电压状态。

④ FRCE（黄色）：有变量被强制时亮。

⑤ RUN（绿色）：CPU 处于运行状态时灯亮。当重新起动 CPU 时，它以 2Hz 的频率闪亮；当 CPU 处于 HOLD 状态时，它以 0.5Hz 的频率闪亮。

⑥ STOP（黄色）：CPU 处于停机状态、HOLD 状态或重新起动时常亮；请求存储器复位时以 0.5Hz 的频率闪亮；正在执行存储器复位时以 2Hz 的频率闪亮。

⑦ BATF（黄色）：电池未装入或失效时灯亮。某些型号的 CPU 上有此 LED 灯，例如 CPU313 和 CPU314 上。

## 5.2 硬件组态的调试

### 5.2.1 下载硬件组态时的调试

下载硬件组态和用户程序以及调试程序的前提是在编程设备和可编程序控制器之间建立合适的接口，例如多点接口 MPI。需要特别注意的是，在第一次下载硬件组态时最好通过 MPI 接口和编程电缆，PG/PC 的设置见图 3－16。根据实际需要，以后的在线连接可以通过 PROFIBUS 接口或通信处理模块等完成。建立在线连接，不仅需要配置适合的硬件接口，而且要正确地设置 STEP 7 中的 PG/PC 接口，二者缺一不可，相互配合。

下载硬件组态时，如果不能正常下载，经常出现系统提示对话框，如图 5-1 或图 5-2 所示。这时出现错误的可能原因及解决方法如下：

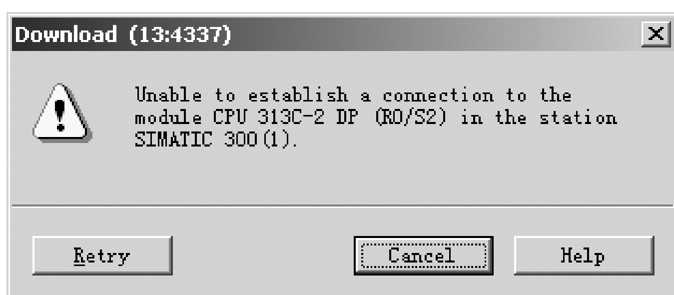



图 5-1 系统提示对话框 a

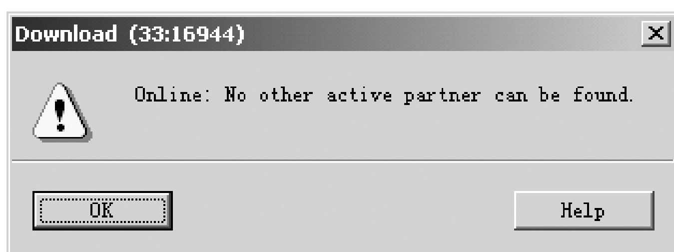



图 5-2 系统提示对话框 b

① 检查是否有物理连接以及这个物理连接是否可靠。

② 检查“Set PG/PC Interface”。一定要根据使用的物理连接选择合适的接口协议，可以参考第 3 章的详细内容。这个错误在网络组态的下载中经常出现，所以下载时思路必须要清晰。

③ 地址可能不一致。比如 PLC 中实际的 MPI 地址是 3，而用户在硬件组态时为 2，二者不一致，当然就不能建立连接了。下载过程中，出现下载硬件时，“Select node address”对话框 a 如图 5-3 所示，点击“View”按钮，在如图 5-4 所示“Select node address”对话框 b 的“Accessible Nodes”项中显示出当前与编程设备连接的 PLC 以及其 MPI 地址，单击这个 PLC 使得“MPI address”项中的 PLC 地址也变为 3，单击“OK”按钮继续下载即可。这时 PLC 的实际 MPI 地址就变为 2，组态下载正确。

若此时还存在如①或②错误，则在图 5-3 中点击“View”按钮后，“Accessible Nodes”项中显示为空，表示没有直接与编程设备连接的 PLC。用户必须先解决①或②的错误。

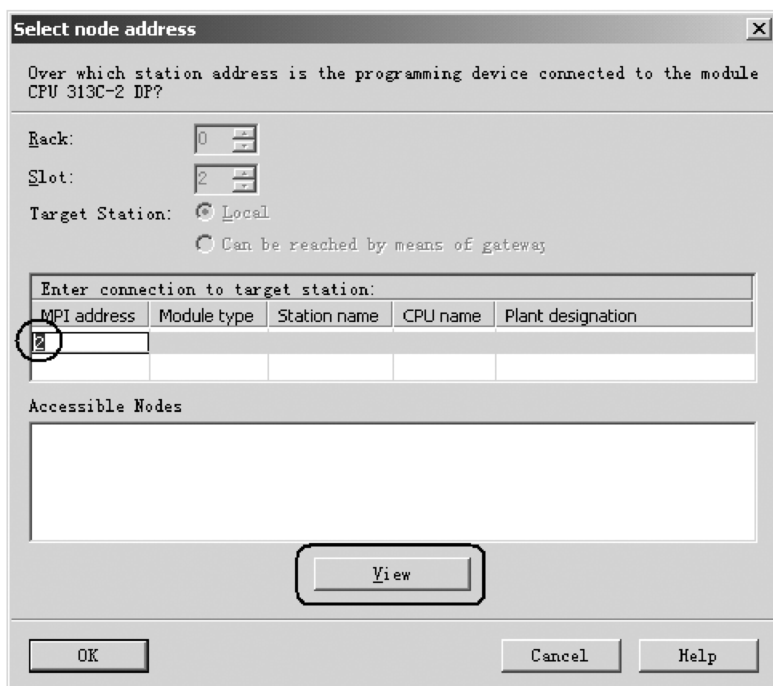



图 5-3 “Select node address”对话框 a

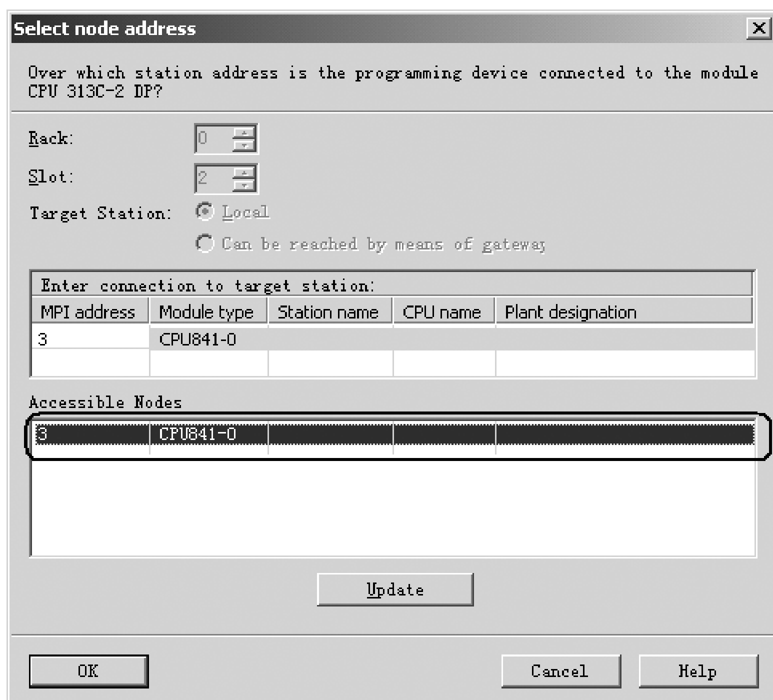



图 5-4 “Select node address”对话框 b

### 5.2.2 建立在线连接

下载硬件组态后，最希望看到的是 CPU 模块上有两个绿灯亮（一个是 DC 5V 指示灯，另一个是 RUN 指示灯）。因为这代表硬件组态正确和通信正常，我们称为“两只眼睛”常亮。

此时，在 SIMATIC Manager 窗口中，用户可以通过“Accessible Nodes”窗口建立在线连接。这种访问方式使用户能快速访问所有与正在使用的编程设备连接的可编程序控制器，可以在线测试网络是否通畅。

具体方法如下：使用菜单命令“PLC/Display Accessible Nodes”或单击工具栏中 按钮，打开“Accessible Nodes -- MPI”窗口，如图 5-5 所示，显示出当前在线的 PLC 及其 MPI 地址，“directly”代表通过编程电缆直接相连的 PLC。用户可以单击图 5-5 左视图中的在线项目结构，打开所有在线的程序块（Blocks）。利用这个工具来测试网络的通信情况尤为便捷。当然，前提是系统的物理网络连接和“Set PG/PC Interface”设置正确。




图 5-5 Accessible Nodes -- MPI 窗口

### 5.2.3 利用“Module Information”工具调试

对于初学者，尤其在做网络的硬件组态时，下载硬件组态后，常常会看到 CPU 模块上有黄灯亮（STOP 指示灯），这代表硬件组态有错误或通信不正常。只有把这个错误处理掉，才能继续其他工作。解决 CPU 停机的故障，应使用“Module Information”工具。在菜单“PLC -> Module Information”中包含了许多检测故障所需要的信息。可以通过 SIMATIC 管理器或程序编辑器进入这一菜单。

在 SIMATIC 管理器中，还可以单击工具栏内的按钮，然后选择在线的 MPI = x（x 是所连接 CPU 的 MPI 地址），接着选择菜单“PLC /Module Information”。如果已在 SIMATIC 管理器中打开了硬盘上的一个项目，选中 S7 项目后可以进入“PLC /Module Information ”菜单。

上电后，CPU 自动检查实际的硬件是否与用户组态的硬件一致或参数设置是否正确。如图 5-6 所示为停机后单击工具栏中按钮，在硬件组态窗口中显示 CPU 的自诊断信息，可以看出 CPU 上的图标以及模拟量模块有错误的图标为 （红色的叉），并且 CPU 上的 STOP 指示灯亮。下面一起来找找原因。

在 SIMATIC 管理器中，通过菜单“PLC/Diagnostic setting/Hardware Diagnostics”，打开在线“Hardware Diagnostics - Quick View”（硬件诊断的快速视窗窗口）对话框，如图 5-7 所示，显示 CPU 处在 STOP 状态，图标为（红色的倒三角形），并且模拟量模块有错误。单击图 5-7 中的“Module Information”按钮，打开有错误模块的详细错误信息“Module Information — AI8 × 12Bit”对话框，如图 5-8 所示，在此可以看出是通道错误。经过详细检查并分析得知：传感器提供的是电流信号，而默认参数设置为电压，二者不一致。所以在图

5-7 中单击“Open Station ONLINE”按钮打开硬件组态窗口，可以重新设置模拟量模块的参数，双击此模块，打开属性窗口，如图 8－20 所示，设置“Measuring Type”为电流信号“4DMU”。保存编译后下载，这时发现 CPU 又重新恢复正常了，通过再打开如图 5-7 所示的对话框，观察到 CPU 上的图标已变为运行图标（蓝色的惊叹号）。

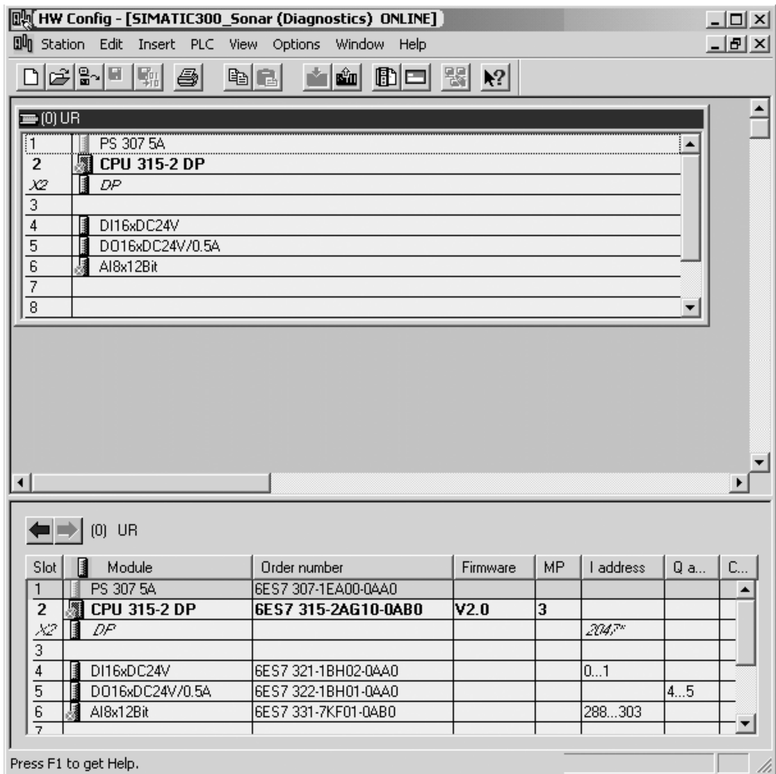



图 5-6　CPU 的自诊断信息

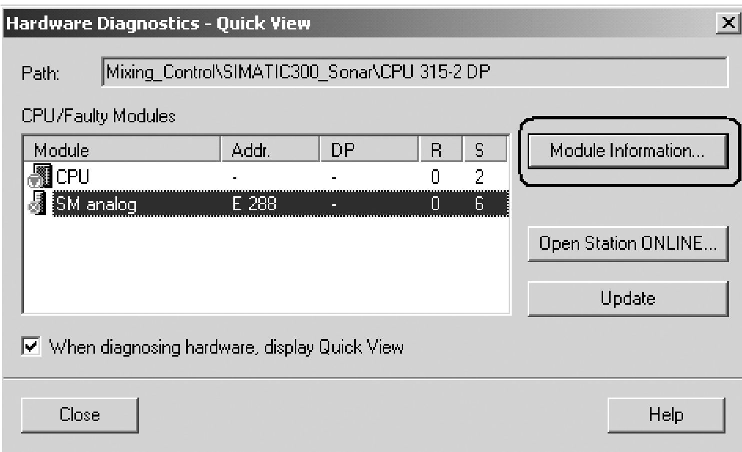



图 5-7　“Hardware Diagnostics－Quick View”对话框

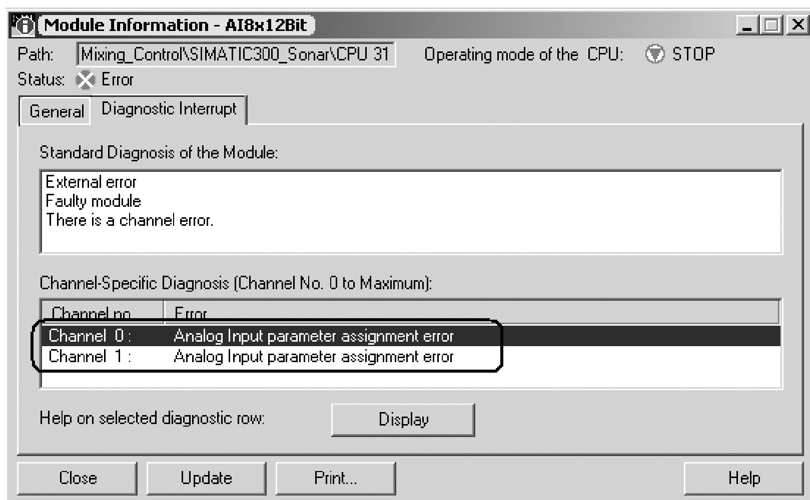



图 5-8 “Module Information — AI8 × 12Bit” 对话框

## 5.2.4 硬件组态窗口中信号的检测与修改

在实际应用中，输入/输出点多达几百个甚至几千个，利用信号检测，可以快捷地得到某个输入/输出点的当前状态。当然，这个功能在程序编辑器和变量表中也可实现。

输入模块的点诊断，在硬件组态窗口中选中该模块，并单击右键选择“Monitor/Modify ”或执行菜单命令“PLC/Modify Variable”打开如图 5-9 所示在线“Monitor/Modify”对话框，选中“Monitor”，此时观察到模块当前的输入输出状态。

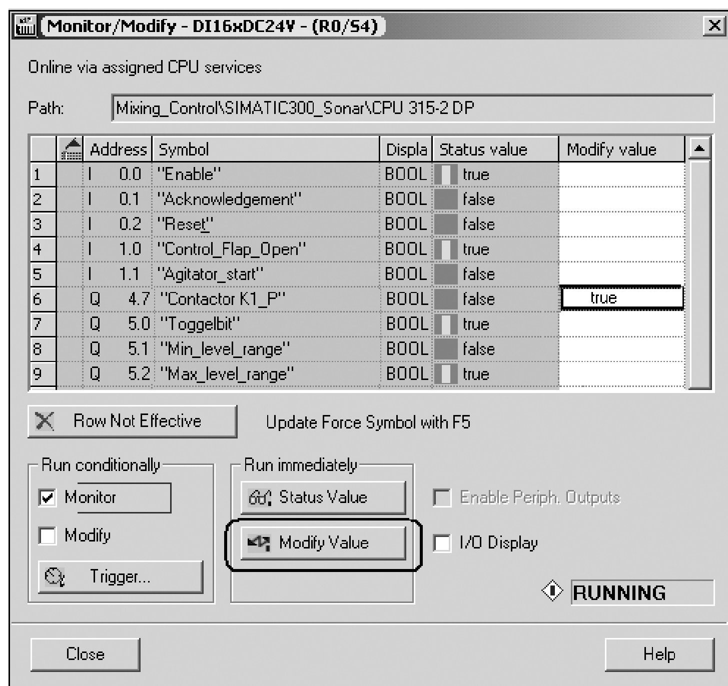



图 5-9 “Monitor/ Modify ” 对话框

对于输出点，除了检测点状态以外，还可以根据实际需求进行修改，在“Modify Value”列中输入将要修改的数值并点击“Modify Value”按钮即可。这个功能可以检测与输出点连接的执行器动作是否正常，当然，在修改前必须确认这样的操作不会引起危险。

### 5.2.5 诊断符号

诊断符号显示于在线项目管理器窗口、在线硬件组态窗口和在线硬件诊断功能打开的快速视窗（见图5-7）中。诊断符号用来形象直观地表示模块的运行模式和模块的故障状态。图5-7 的打开方法为：在 SIMATIC 管理器中，执行菜单“PLC/Diagnostic setting/Hardware Diagnostics”。诊断符号可使故障检查更直观。如果模块有诊断信息，则在模块的符号上会增加一个诊断符号，或显示的模块符号对比度降低。

常用的诊断符号及其意义如下：

当前组态与实际组态不匹配：被组态的模板不存在，或插入了不同类型的模板。

故障：模板出现故障可能的原因，包括诊断中断、I/O 访问错误或检查到故障 LED 亮。

无法诊断：可能无在线连接或该 CPU 不支持模板诊断信息。

停机（STOP）：故障引起的停机，或 CPU 面板的模式开关拨至 STOP 位置。

运行（RUN）。正常运行。

保持（HOLD）：一般处在单步与断点调试时。

强制与运行：表示在该模板上有变量被强制，即在模板的用户程序中有变量被赋予一个固定值，该数据值不能被程序改变。

## 5.3 离线/在线程序块的比较

下载程序后，在 SIMATIC 管理器中，点击“Blocks”，使用菜单命令“View/Online”或单击工具栏中按钮，打开项目的在线视图，建立整个程序块的在线连接，其标题栏由项目名和“ON LINE”组成，并且底色用浅蓝色显示。单击工具栏中按钮，显示离线视图。用户可以通过菜单“Windows/Arrange”合理安排两个窗口，如图5-10 所示。

我们可以根据图5-10 比较项目的在线/离线视图，确认 PLC 中的程序块与编程器中的一致。离线视图显示出编程器中存储的项目内容，此时只有 OB1。在线视图显示出当前在线 CPU 中的实际内容，可以看到不仅包括 OB1，还包括 OB82、OB86、OB122 以及 SFC 和 SFB。其中，SFC 和 SFB 是已预存储在每个 CPU 的系统块，而 OB82、OB86 和 OB122 可能是 CPU 旧项目中遗留的组织块。通过菜单“Option/Compare Blocks”比较离线的块与在线的块是否相同，如图5-11 所示，在弹出的对话框中选择 ONLINE/Offline，并单击 Compare 按钮进行比较，弹出离线的块与在线的块的比较信息对话框如图5-12 所示，OB82、OB86 和 OB122 只存在于在线块中，说明它们并不是用户程序的一部分。为了避免引起误操作，最好在在线视图中，将这三个块删除。

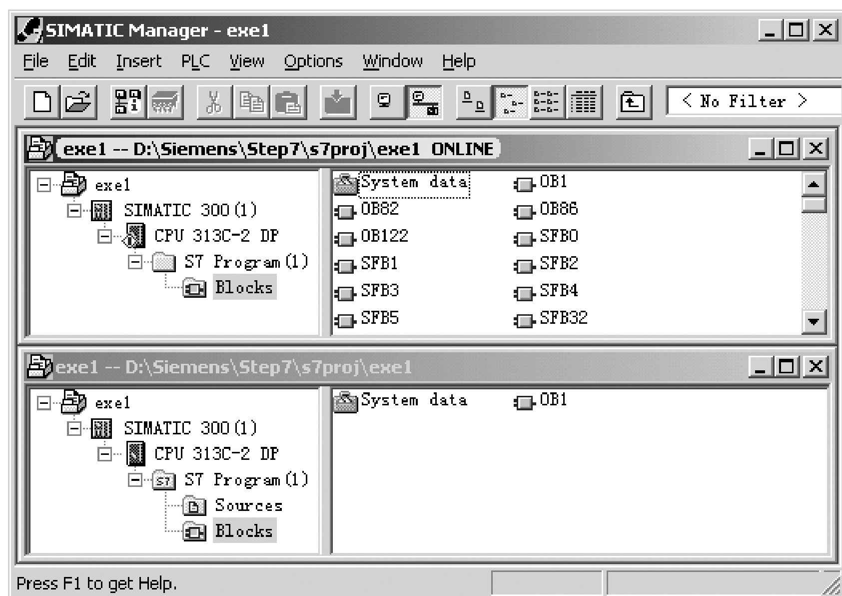



图 5-10　SIMATIC 管理器中的在线/离线视图

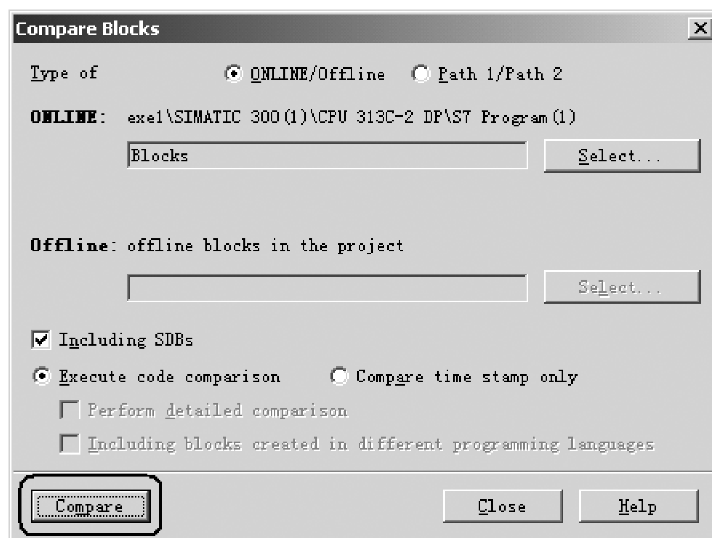



图 5-11　离线的块与在线的块的比较

| Block | Result of comparison |
|---|---|
| OB82 | only exists in Path 1 ONLINE |
| OB86 | only exists in Path 1 ONLINE |
| OB122 | only exists in Path 1 ONLINE |

图 5-12　离线的块与在线的块的比较信息

注意：有时用户的 CPU 会出现罢工现象，SF 和 STOP 指示灯亮，此时使用菜单命令“PLC/Clear and Reset”也不能将故障清除，这时必须在线时将不需要的程序块删除。这样又会看到两只眼睛——CPU 又恢复正常了。

在 STEP 7 中，用户不仅可以在 SIMATIC Manager 中建立在线视图，而且在编程窗口、数据查看窗口和硬件组态窗口中，都可以通过工具栏中的相应按钮建立在线连接。

## 5.4　利用程序状态调试

### 5.4.1　监控程序状态的前提

要显示程序的状态，必须满足下列要求：

① 必须保存编译了的正确程序，并且下载到 CPU。

② 将 CPU 模式开关拨到 RUN 或 RUN - P 位置，即保证用户程序在执行状态。

③ 要监控的程序块必须在线打开。

建议用户在调试程序时，应该在 OB1 中一次调用一个块单独调试，最后再调用整个程序进行综合调试。

### 5.4.2　监视程序的状态

在程序编辑器窗口中，单击工具栏中按钮，进入程序的监视状态，当前窗口中显示在线的程序运行情况。用户可以根据其控制目的，查找程序中的逻辑错误。对于不同语言编写的程序，其监视界面也各不相同。

在梯形图程序监视状态（见图 5-13）和功能块图程序监视状态（见图 5-14）中，用不同颜色、不同形式的线条显示出信号流的状态。当状态满足时用绿色实线表示，状态不满足时用蓝色虚线表示。而且，在变量位置显示该变量的当前值。使用菜单命令“Options/Customize”，在打开的对话框中选择“LAD/STL”选项页可改变监控程序的颜色及线型。

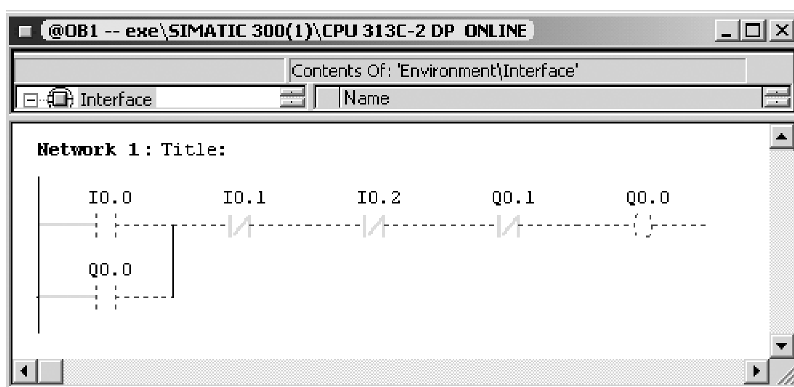



图 5-13　梯形图程序监视状态

在语句表程序监视状态（见图 5-15）中，左视图显示 STL 程序，右视图显示每条执行后的逻辑运算结果（RLO）、状态位（STA）和累加器 1（STANDARD）。通过执行菜单命令

"Options/Customize"，在打开的对话框中选择"STL"选项页可增减需要监视的内容。

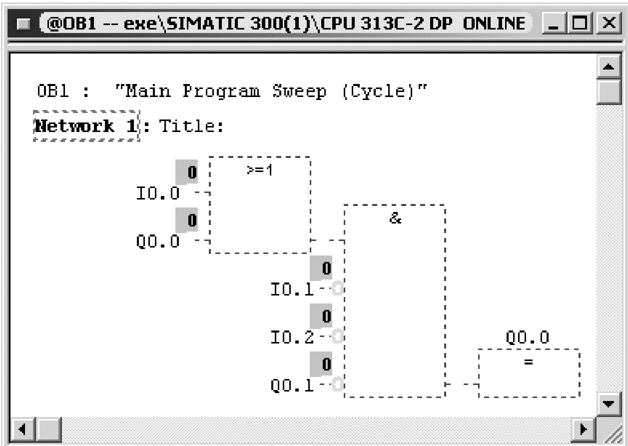



图 5-14　功能块图程序监视状态

@OB1 -- exe\SIMATIC 300(1)\CPU 313C-2 DP ONLINE

Contents Of: 'Environment\Interface'

Interface　Name

OB1 : "Main Program Sweep (

Network 1: Title:

| STL | | | RLO | STA | STANDARD |
|---|---|---|---|---|---|
| A( | | | 0 | 1 | 0 |
| O | I | 0.0 | 0 | 0 | 0 |
| O | Q | 0.0 | 0 | 0 | 0 |
| ) | | | 0 | 1 | 0 |
| AN | I | 0.1 | 0 | 0 | 0 |
| AN | I | 0.2 | 0 | 0 | 0 |
| AN | Q | 0.1 | 0 | 0 | 0 |
| = | Q | 0.0 | 0 | 0 | 0 |

图 5-15　语句表程序监视状态

## 5.4.3　STL 程序的单步与断点调试

只有在 STL 程序中才能使用单步与断点调试，所以对于用其他语言编写的程序，可以通过菜单"View/STL"，将当前视窗内容改变为 STL 程序。单步与断点是调试程序的有力工具，有这种功能的 PLC 并不多见。CPU 是否支持断点和支持断点的数目都与其型号有关。

在单步模式下，CPU 一次只执行一条指令。

### 1. 设置断点与进入单步模式的条件

① 在程序编辑器中，执行菜单命令"Options/Customize"，在打开的对话框中选择"STL"选项页，激活"Active new breakpoints immediately"（立即激活新断点）选项。

② CPU 必须工作在测试模式，执行菜单命令"Debug/Operation"，选择"Test Operation"（测试操作）。

③ 调试的程序必须在线，单击工具栏中按钮即可。若程序在线时又被修改，则必须

重新下载到 CPU 中。

注意：设置断点和起动程序状态监控功能是两种测试方法，二者不能混用，只能根据实际需要选择其一。在 STL 程序中，有断点的行、调用块的参数所在的行、空的行或注释行等位置不能设置断点。

2. 设置断点与单步操作

在 STL 程序中，将光标放在要设置断点的指令行上，执行菜单命令“Debug/Set Breakpoint”或单击工具栏中按钮后，在选中的语句左边出现一个紫色的空心小圆○，表示断点设置成功。如果菜单命令“Debug/Breakpoints Active”处于激活状态（选项前有“√”），则执行菜单命令“Debug/Set Breakpoint”后，将出现一个带有向右的黄色箭头的紫色实心小圆，并且同时出现一个显示 PLC 寄存器内容的窗口，如图 5-16 所示为断点及断点处 PLC 寄存器中的内容。需要注意的是，要使断点起作用，必须执行这个菜单命令来激活断点。这时，CPU 进入 HOLD（保持）模式，在状态栏中以黄色底色显示。在 HOLD 模式下，CPU 不处理指令，所有定时器被冻结。

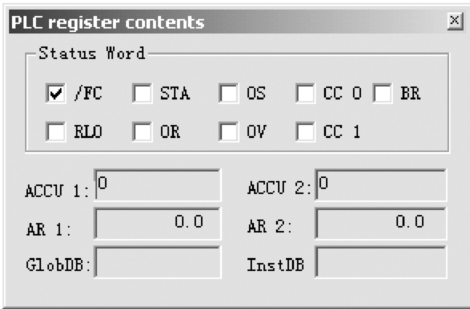


图 5-16　断点及断点处 PLC 寄存器中的内容

紫色实心圆中的黄色箭头代表指向当前执行的指令，执行菜单命令“Debug/Execute Next Statement”，或单击工具栏中按钮后，黄色箭头移动到下一条语句，表示单步模式执行指令。如果下一条是调用程序块的语句，那么通过执行菜单命令“Debug/Execute Call”进入被调用的块，并且可以进行块内程序的调试，块结束时将返回块调用语句的下一条语句。

执行菜单命令“Debug/Resume”或单击工具栏中按钮，可以使程序直接运行到下一个断点。若程序中只有这一个断点，则执行此命令相当于使程序循环执行一次后，又回到断点的位置。若将光标置于断点所在的行，则执行菜单命令“Debug/Insert Breakpoints”或单击工具栏中按钮，可以删除此行的断点。执行菜单命令“Debug/Show Next Point”或单击工具栏中按钮，使光标跳到下一个断点。执行菜单命令“Debug/Execute Call”或单击工具栏中按钮，可以进入块调用的内部执行。断点调试的按钮设置与一般编程语言的设置相同，比较容易理解。

## 5.5　利用变量表调试

### 5.5.1　变量表的功能

编程器和实际 PLC（或 PLCSIM）建立在线联系后，可以将硬件组态和程序下载到实际 PLC（或 PLCSIM）中。用户可以通过 STEP 7 进行在线调试程序，寻找并发现程序设计中的问题。如果程序较大，那么用户在屏幕上就不能同时观察调试过程中变量的变化过程。为了解决这个问题，可以建立变量表。使用变量表可以在一个画面上同时显示用户感兴趣的全部

变量。变量表是用于监视和修改变量值的一个重要的调试工具。

利用变量表进行调试程序时，有以下功能：

① 监视变量：可以在编程设备上显示用户程序或 CPU 中每个变量的当前值。

② 修改变量：可以将固定值赋给用户程序或 CPU 中的每个变量，使用程序状态测试功能时也能立即进行一次数值修改。

③ 使用外设输出并激活修改值：允许在停机状态下将固定值赋给 CPU 中的每个 I/O。

④ 强制变量：可以为用户程序或 CPU 中的每个变量赋予一个固定值，这个值是不能被用户程序覆盖的。

用户可以赋值或显示数值的变量包括：输入、输出、位存储、定时器、计数器、数据块的内容和 I/O（外设）。用户可以通过定义触发点和触发频率来决定变量什么时候、以什么样的频率被监视或赋予新值。

## 5.5.2　建立变量表

利用变量表调试程序之前，必须生成一个变量表并输入需要监视的变量。建立变量表可以选择如下三种方法：

① 一般在 SIMATIC Manager 中生成变量表，选择“Blocks”文件夹，使用菜单命令“Insert/S7 Block/Variable Table”，或在右视图中单击右键使用菜单命令“Insert New Object/ Variable Table”，打开变量表的属性对话框，可以为新建的变量表命名，如 VAT－1，单击“OK”按钮后建立一个新的变量表，如图 5-17 所示为变量表的图标。通过双击这个对象即可打开新建的变量表。

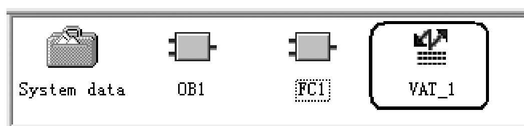



图 5-17　变量表的图标

② 在程序编辑窗口中，执行菜单命令“PLC/Monitor/Modify Variables”，直接生成一个无名的变量表，输入需要监视或修改的变量后，单击变量表视窗中的保存按钮，可以在打开的保存对话框中为这个变量表命名，并选择保存在项目路径的“Blocks”下。保存好新建的变量表后，同时也可在 SIMATIC 管理器中显示，如图 5-17 所示 。

③ 在打开的 Variables Table（变量表）编辑器中，执行菜单命令“Table/New”生成一个新的变量表，该表没有被赋给任何程序，所以需要按照上述第二种方法进行保存。

一旦生成一个变量表，用户可以存储、打印和多次使用该表进行调试工作，但变量表并不下载到 PLC 中。

## 5.5.3　变量表的使用

### 1. 输入变量

每个变量表中有五栏，分别显示每个变量的五个属性：地址（Address）、符号（Symbol）、显示格式（Monitor Format）、状态值（Status Value）和修改值（Modify Value）。一个变量表最多可有 1024 行，每行最多可有 255 个字符。

用户可以通过在“符号”栏输入符号，或在“地址”栏输入地址来插入变量，如果在符号表中已经定义地址相应的符号，则符号栏或地址栏会自动填入。如图 5-18 所示为建立变量表，输入将要监控的变量。在显示格式栏中，可以单击右键选择即将在状态值栏和修改

值栏中变量的显示格式。使用菜单命令“Insert/Range of Variables”，打开“插入变量的范围(Insert Range of Variables)”对话框，用户可以插入多个地址连续的变量。

输入变量时需要注意以下几点：

① 只能输入已在符号表中定义过的符号。

② 将各个逻辑块中相关联的变量集中输入。

③ 如果符号名中含有特殊字符，则必须用引号括起来（例如，“Motor. off”、“Motor + Off”、“Motor - off”）。

当变量表中输入变量时，在每行的结束都会执行语法检查。任何不正确的输入都会被标为红色。

若建立注释行，则在输入时以“//”开始，那么此行将变为绿色。

用户如果想使一行或多行变量无效，可以先选中一行或多行变量，然后使用菜单命令“Edit/Row without Effective”或工具栏的按钮。这种方法也可以建立一个注释行。

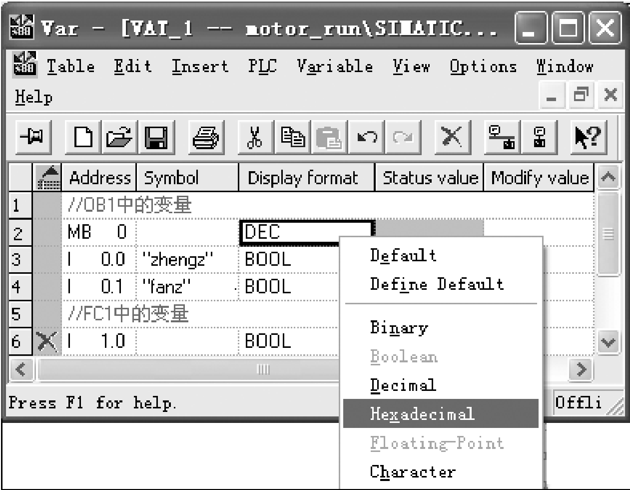



图 5-18　建立变量表

### 2. 建立与 CPU 的连接

使用变量表监视和修改变量的前提是必须要与 CPU 建立在线连接。

如果有在线连接存在，变量表窗口的标题栏中会显示“ONLINE（在线）”。在状态栏中，根据 CPU 的操作状态，将显示“RUN”、“STOP”、“DISCONNECTED”或“CONNECTED”。

如果与所要求的 CPU 的在线连接不存在，使用菜单命令“PLC/Connect To/…”来定义与 CPU 的连接，以便进行变量的监视或修改。这个菜单有三个子菜单选项，分别如下：

① Configured CPU：与点击工具栏中按键作用相同，用于建立被激活的变量表与 CPU 的在线连接。

② Direct CPU：与点击工具栏中按键作用相同，用于建立与直接连接的 CPU 之间的在线连接。直接连接的 CPU 指与编程设备用编程电缆连接的 CPU。

③ Accessible CPU：执行这个选项后，在打开的对话框中，用户可以选择与哪个 CPU 建立连接。如果已经与一个 CPU 建立了连接，那么使用这个命令可以选择与另一个 CPU 建立连接。系统支持一个变量表与不同的 CPU 建立连接。这种连接方便了调试。

使用菜单命令“PLC/Disconnect”，可以中断变量表和 CPU 的连接。

### 3. 触发设置

在调试程序过程中，用户有时需要监视变量在某一特定点（触发点）的当前数值，以便更明确地掌握程序的运行过程。使用菜单命令“Variable/Trigger”，或点击工具栏中按钮，打开如图 5-19 所示变量表的触发设置对话框，可以设置触发点和触发条件。

触发点是监视的变量将要显示数值的时间点，有三种选择方式：在扫描循环开始时触发、在扫描循环结束时触发和 CPU 工作状态从 RUN 转为 STOP 时触发。触发条件可以选择只触发一次或者每个循环周期都触发。一般情况下的监控，用户使用默认设置即可。

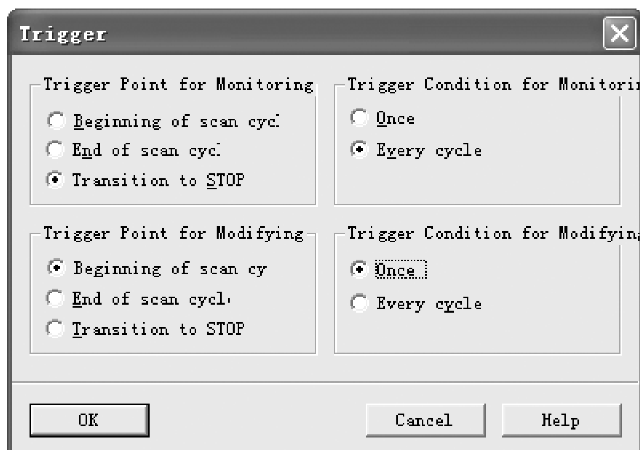



图 5-19　变量表的触发设置对话框

### 4. 监视变量

完成上述 1、2 和 3 三个步骤后，就可以准备起动监视变量功能，其目的是直接在编程器上观察到程序执行的状态。首先 CPU 的模式开关拨到 RUN－P 位置，使用菜单命令“Variable/Monitor”，或单击工具栏中按钮，我们常称为“戴眼镜”，起动变量监视功能。这时在状态值栏中显示出 CPU 运行中当前的变量值。

可以用菜单命令“Variable/Monitor”或再次单击工具栏中按钮关闭监视功能。使用菜单命令“Variable/Update Monitor Values”或单击工具栏中按钮，可以对所选变量的数值做一次立即刷新。如果在监视功能激活的状态下按〈ESC〉键，则不经询问就退出该功能。

### 5. 修改变量

修改变量主要针对与程序有关的 M 区变量和 DB 区变量（作为触点）的改变。因为在 RUN（或 RUN－P）模式下，如果数字量输出变量受到程序的控制输出为 0，用户则不能随意改变程序运行的结果，那么就不能在变量表中将其修改为 1。在 RUN（或 RUN－P）模式下同样也不能改变数字量输入（I 映像区）的状态，因为它们的状态取决于外部电路的通/断。

在 STOP 模式下，因为没有执行程序，各变量的状态是独立的，所以修改变量不受限制。I、Q 和 M 区的数字量变量都可以任意设置为 1 状态或 0 状态，并且可以保持，相当于对它们置位或复位。这个特殊功能常用来测试数字量输出点的硬件功能是否正常。

修改变量的方法如下：首先起动监视变量功能，随时观察变量值，然后在变量表中的修改值（Modify Value）栏中输入新的变量值，执行菜单命令“Variable/Modify”，或单击工具栏中按

钮激活修改功能，将修改值立即送入 CPU，从而改变程序的执行。用户可以用菜单命令“Variable/Activate Modify Values”，或单击工具栏中按钮对所选变量的修改数据作一次立即刷新。

当选中 Modify Value 栏中的变量修改值时，单击工具栏中按钮，可以使该变量的修改值暂时失效。此时，变量修改值会以“//”起始作为注释。再按下这个按钮，可以使修改值重新生效。

6. 强制变量

执行强制变量命令可以给用户程序的变量赋一个固定值，它独立于程序的运行，不会被 CPU 中正在执行的用户程序改变或覆盖。强制优点在于可以在不用改变程序代码，也不用改变硬件连线的情况下，强行改变输入和输出的状态。所以，用户可以为程序设置特定的值并用该方法对已编程的功能进行测试。实现这一功能的前提是 CPU 支持该功能。

选中将要强制的变量，执行菜单命令“Variable /Display Force Values”后，强制数值窗口处于激活状态。然后在强制变量窗口如图 5-20 所示的“Force Value”列中，输入强制的数值，执行菜单命令“Variable /Force”进行变量的强制，此时激活的强制变量（以红色的 F 标记）和它们的强制值就都显示在窗口中了。

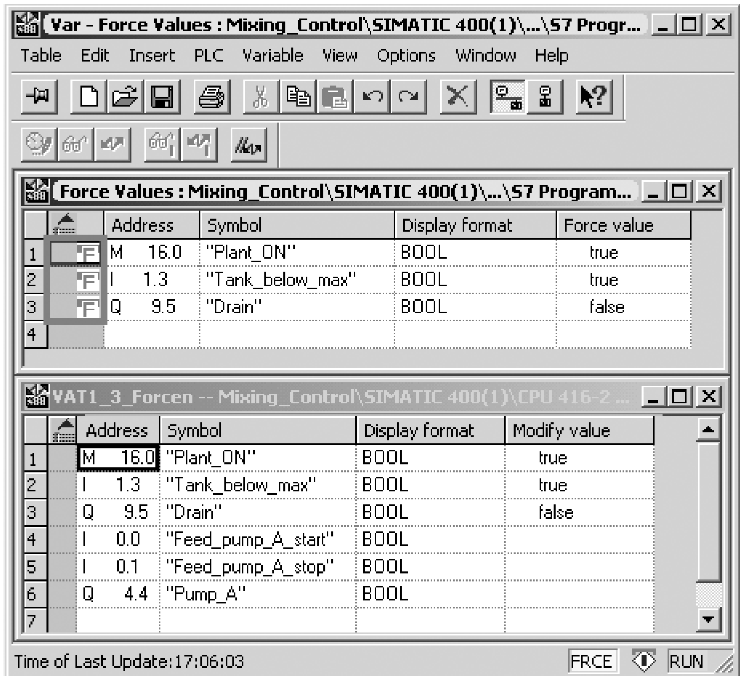



图 5-20　强制变量窗口

## 5.6　利用“诊断缓冲区”调试

故障诊断是指 PLC 内部集成的错误识别和记录功能，S7 -300/400CPU 和其他模块具有强大的故障诊断能力，配合 STEP 7 软件的使用，用户可以获得大量的硬件故障和编程错误的信息，使用户能够迅速查找故障及排除故障。

记录错误信息的区域称为诊断缓冲区。诊断缓冲区是存放在 CPU 中的一个先进先出区域，它由后备电池来保持，对存储器的复位也不能清除该缓冲区的内容。它存储按照时间发生顺序排列的诊断事件，而且所有的事件也可以在编程器上按照它们出现的顺序进行显示。这个区域的大小有赖于 CPU 型号，例如 CPU 314 可存储 100 条信息。如果缓冲区满，则最旧的信息将被覆盖。

诊断缓冲区是“Module Information”工具的一部分。它可以通过 SIMATIC 管理器的菜单“PLC/Diagnostic setting /Module Information/Diagnostic Buffer”或从程序编辑器的菜单“PLC /Module Information/Diagnostic Buffer”进行访问。

利用 CPU 的诊断功能，可以识别 CPU 或模块中的系统错误和 CPU 中的程序错误。如果必要，将自动激活一个相关的异步错误组织块或同步错误组织块（详细信息参见第 8 章）。

如果一个事件发生时，例如，CPU 的操作模式由 STOP 到 RUN 转换，那么标有时间和日期的信息被保存到诊断缓冲区中，如图 5-21 所示为“Diagnostic Buffer”中的内容 1。诊断缓冲区显示了三条最新的信息，点击每一条信息，在窗口下方的“Details”区域会同时显示详细的诊断内容，综合这三条信息可知这个事件使 CPU 的操作模式改变。

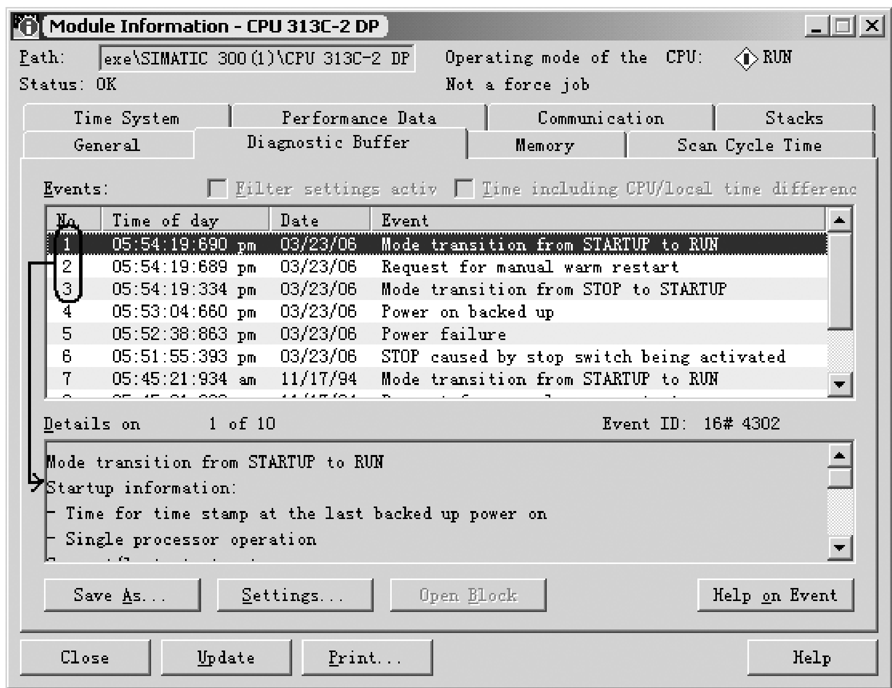



图 5-21 “Diagnostic Buffer”中的内容 1

如果导致故障的原因是程序的错误，例如在 OB1 中调用了 FC6 逻辑块，而只下载了 OB1 没有下载 FC6。则在下载后 CPU 操作模式由 RUN 转为 STOP，模块面板上的 SF 和 STOP 灯亮。打开“Diagnostic Buffer”中的内容 2，如图 5-22 所示，出现两条信息，第一条显示 OB1 中编程错误，第二条明确指出 FC6 没有被下载。单击“Open Block”按钮可以在线打开有错误的块 OB1，并且直接以红色显示错误的段。在 STL 方式下，光标将停在发生错误的语句之前。

将 FC6 下载到 CPU 中，但是 CPU 操作模式仍为 STOP，此时需要重新起动。通过 SIMATIC 管理器的菜单“PLC/Diagnostic setting /Operation Mode…”打开“Operation Mode”对

话框，如图5-23所示，单击“Warm Restart”，或者拨模式开关至STOP，再拨回RUN位置。这时，观察到CPU模块上的指示灯恢复正常。

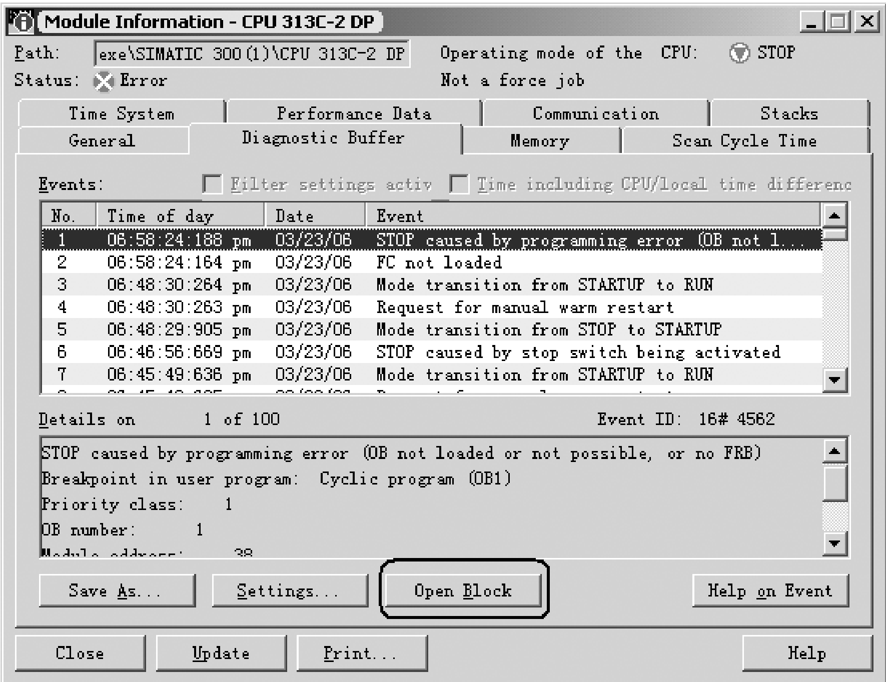



图5-22 “Diagnostic Buffer”中的内容2

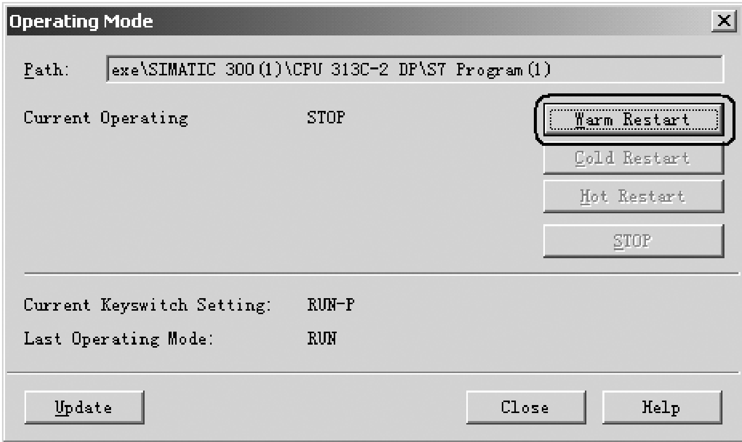



图5-23 “Operating Mode”对话框

并不是每个故障或错误都能在诊断缓冲区中找到原因。针对下列故障，应采用不同的手段予以排除：

① 导致CPU停机的故障，应使用“Module Information”工具。

② 逻辑错误，即程序可执行但功能不能实现，应使用变量表和程序状态工具。

③ 偶尔出现的故障，即只在特定的系统状态下才出现的故障，它可能导致停机或逻辑错误。可采用“CPU Messages”工具。

注意：在使用诊断缓冲区内容时，最好事先为 CPU 校对时间，便于用户根据时间信息调试。通过 SIMATIC 管理器的菜单“PLC/Diagnostic setting / Set Time of Day”或从程序编辑器的菜单“PLC /Set Time of Day”进行访问，打开如图 5-24 所示的“Set Time of Day”对话框。“PG/PC time”栏显示的是当前编程器的时间，“Module Time”栏显示的是 CPU 模块的时间，激活“Take from PG/PC”复选框，再单击“Apply”按钮，就可以将模块的时间校对为编程器的时间。

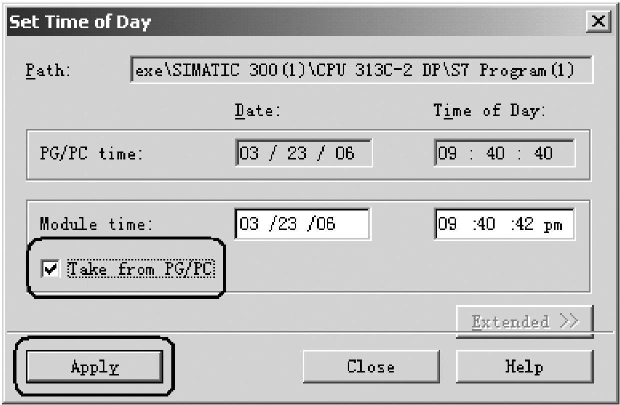



图 5-24　“Set Time of Day”对话框

在其他选项页中，可以查看系统的在线信息。如图 5-25 所示为“Module Information”——Memory 在线信息。装载存储器（Load memory）集成在 CPU 的 RAM 中，它的使用情况用左边的棒图显示。如果在 CPU 上插入存储卡，它的使用情况用中间的棒图显示。附加信息也存储在装载存储器中，所以装载存储器总是大于工作存储器。工作存储器（Work memory）只存储与 CPU 中程序运行有关的数据。

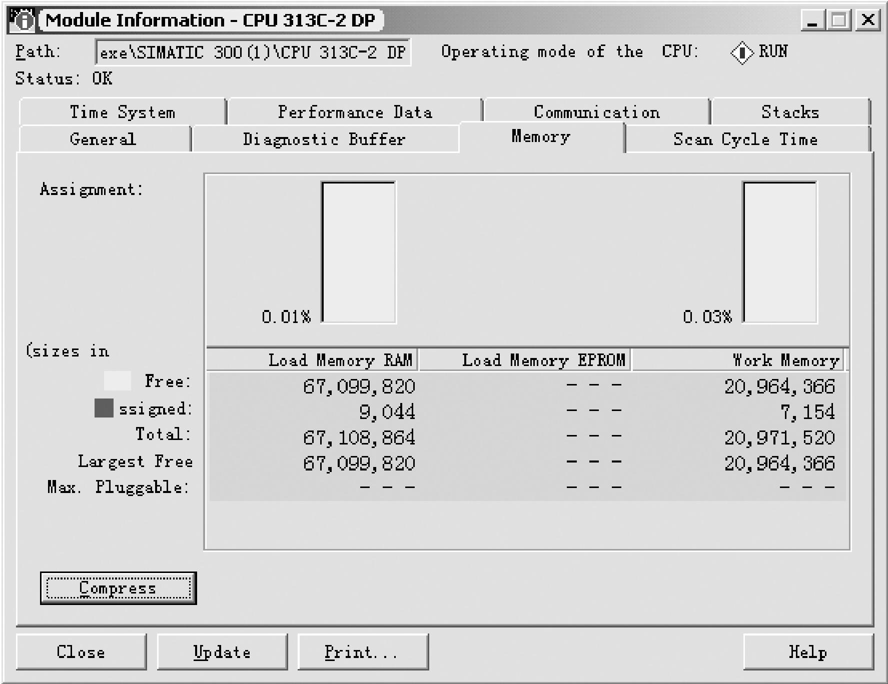



图 5-25　“Module Information”——Memory 在线信息

## 5.7 参考数据（Reference Data）

对于排除逻辑错误，“Program Status”（程序状态）和“Reference Data”（参考数据）是两个非常有力的工具。例如，在监视程序状态时发现一个内存位的条件不成立，可以利用参考数据工具来确定该位是在哪里被设置的。对地址的多次赋值是一种常见的错误，也就是该地址在程序的多处被赋值，利用参考数据工具可以很容易地发现这类错误。

参考数据通过直观的表格方式显示，可以让用户对程序的调用结构、资源占用情况等一目了然。利用参考数据，用户调试和修改程序会更加方便。

### 5.7.1 参考数据的生成和显示方式

用户编好程序后，在 SIMATIC 管理器中，选择要生成参考数据的块文件夹，然后通过菜单命令“Options/Reference Data/Generate”生成参考数据。如果用户在修改程序后又要重新生成参考数据，计算机会提醒用户是否刷新或重新生成。

只要通过菜单命令显示参考数据表，打开后也无需保存。

STEP 7 中可显示五类参考数据。显示参考数据的方法：

① 从 SIMATIC 管理器中显示：选择“Blocks”文件夹，选择菜单命令“Options/Reference Data/Display”。

② 从编程语言编辑器窗口显示：选择菜单命令“Options/Reference Data/Display”。

### 5.7.2 参考数据表的种类

执行菜单命令后，系统提示选择生成参考数据表的种类，如图 5-26 所示，五类参考数据表分别如下：

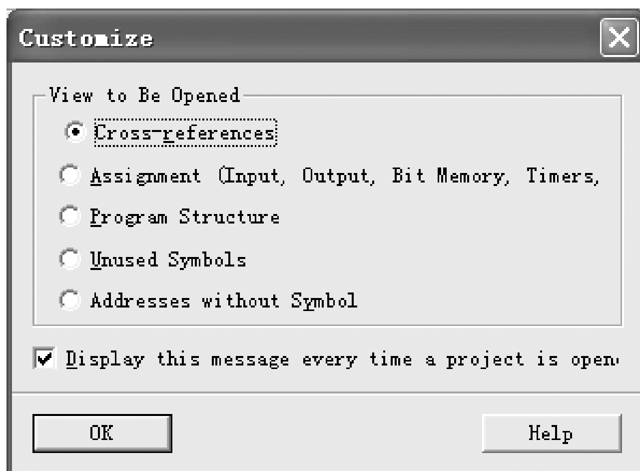



图 5-26　选择生成参考数据表的种类

1. 交叉参考表（Cross Reference）

交叉参考表给出了用户程序中所用地址的概述，如图 5-27 所示为交叉参考表窗口。

利用交叉参考列表，可以得到输入（I）、输出（Q）、位存储（M），定时器（T）、计数器（C）、功能块（FB）、功能（FC）、系统功能块（SFB）、系统功能（SFC）、PI/ PO 和数据块（DB）。这些存储区域中被用户程序使用的地址列表，详细显示了绝对地址和符号地址及使用情况。

交叉参考列表的默认选项是按存储区域分类。如果用鼠标点中栏标题，则可以按默认分类标准对这一栏的输入项进行分类。

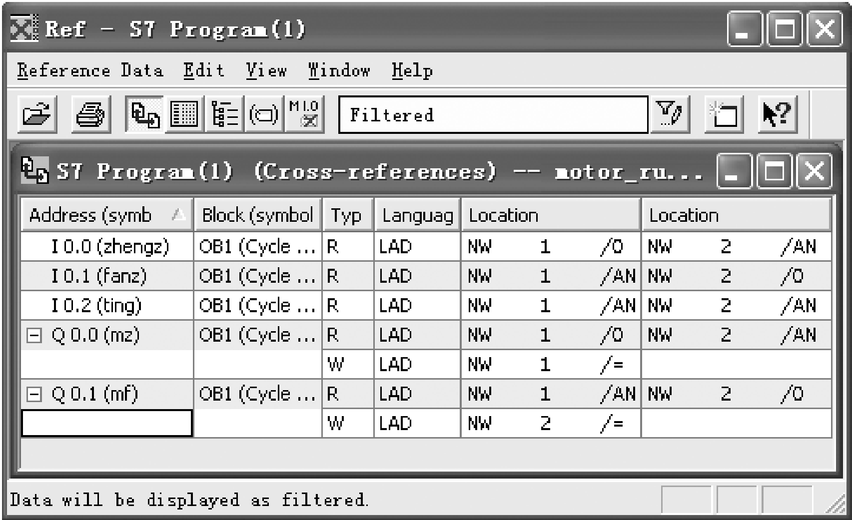



图 5-27　交叉参考表窗口

2. 赋值表（Assignment）

如图 5-28 所示，赋值表显示在用户程序中已经赋值的地址，使用户能概括地了解输入（I），输出（Q）、位存储（M）、定时器（T）和计数器（C）中哪个字节中的哪一位被使用了。图中每一行包含存储区的一个字节，在该字节中有八个位，按其访问的情况标有代码，如表 5-2 所示为赋值表中的代码含义，并指示访问是否为一个字节、一个字或一个双字。这是用户程序的故障查询和修改的重要基础。

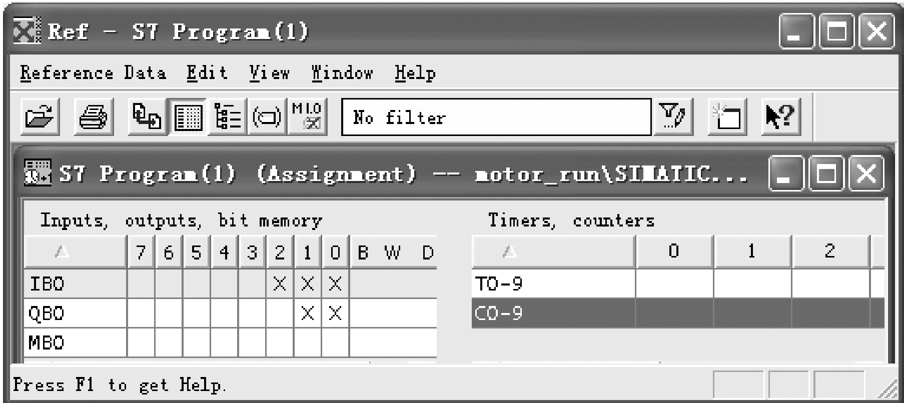



图 5-28　赋值表

表 5-2　赋值表中的代码含义

| | |
|---|---|
| 白色背景 | 地址未被访问，因而也没有赋值 |
| X | 直接访问的地址 |
| 蓝色背景 | 间接访问的地址（字节、字或双字访问） |

3. 程序结构（Program Structure）

程序结构显示了在用户程序中程序块的分层调用结构。可以对程序中所用的块、它们之间的从属关系以及它们对局域数据的需求有一个概括性的了解。

4. 未使用的符号（Unused Symbols）

显示已经在符号表中定义，却未在用户程序的任何一部分中使用的符号。表中的每一行由符号、地址、数据类型和注释组成。

5. 没有符号的地址（Address Without Symbols）

显示 S7 用户程序中已经使用、却未在符号表中定义的地址的列表。表中的每一行包括地址和该地址在用户程序中使用的次数。

### 5.7.3　在程序中快速查找地址的位置

编程时使用参考数据，可将光标定位于某一个地址在程序中的不同位置上。使用这个功能的前提是必须生成最新的参考数据，但不需要显示这个参考数据表。具体方法如下：

1）在编程语言编辑器窗口中，选中打开块的一个地址。

2）选择菜单命令“Edit/Go To/Location”或单击右键选择命令“Go To/Location”，如图 5-29 所示为快速查找地址对话框，图中显示该地址在程序中的位置列表。

3）选择选项“Overlapping access to memory areas（地址区域的重叠访问）”，则显示与被调用的地址的物理地址或地址区域相重叠的那些地址的位置，重叠“地址”栏被加到表中。

4）在列表中选中某栏并点击“Go To”按钮，画面就回到编辑器窗口中，光标将定位于该地址所在的网络。

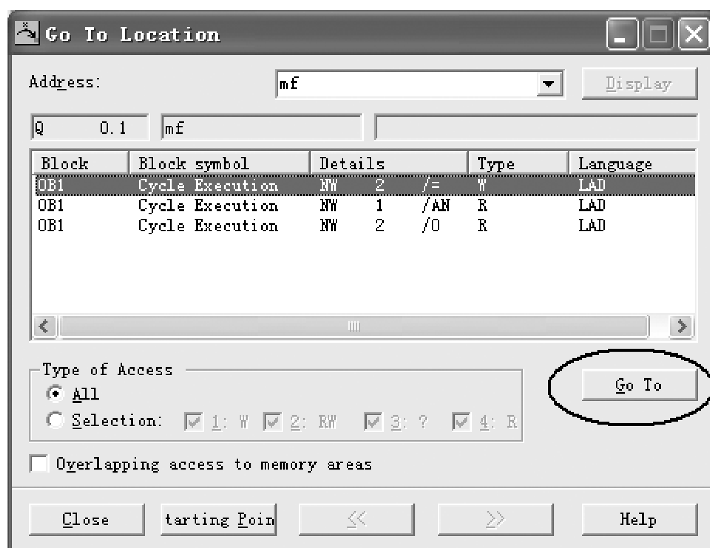



图 5-29　快速查找地址对话框

## 5.8 结构化程序的调试

结构化编程简化了程序的组织，使程序的可读性增强。更重要的是，由于可以分别测试每一个结构程序块，所以查错、调试和修改都更容易。

例如，一个结构化程序的结构示意图及其调试顺序如图5-30所示。调试结构化程序的方法是按顺序一步一步地调试每一个程序块。首先下装除组织块外的用户程序。然后按下述步骤调试：

1）第一步是通过下装起动组织块（OB100～OB102）来测试起动特性。

2）从嵌套最深的块（例如：FB4）开始调试。需要下装FB4，并且在OB1中插入一个块调用指令（CALL FB4）并下载OB1，再一步一步地测试循环程序。

3）然后，调试功能块，它包括一组块（FC3、FC2和FC1）。为此，在OB1中插入一个带有BEU指令的段。当所有的程序都被调用后，再删除这个段。

4）根据程序的结构，用于中断处理的程序在最后测试（如果该中断程序不影响程序的循环执行），或在循环程序的测试过程中调试。

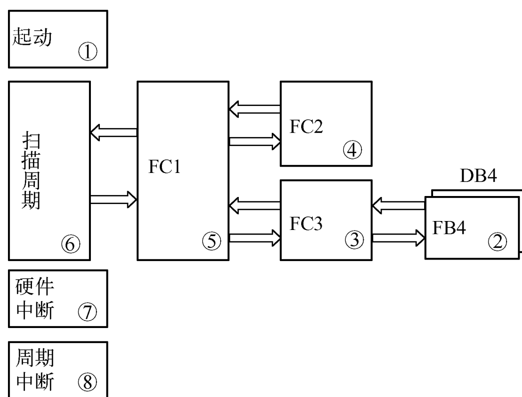



图5-30 结构化程序的结构示意图及其调试顺序

## 5.9 S7－PLCSIM的应用

### 5.9.1 S7－PLCSIM介绍

S7－PLCSIM是自动嵌套在STEP 7中的一个非常实用的软件。我们可以把它作为一台仿真的PLC，用于运行和测试用户程序。因为这种仿真完全是用STEP 7软件进行的，因此无需连接任何S7硬件，就可以在PG/PC上仿真一个完整的S7－CPU，包括地址和I/O。S7－PLCSIM使用户能够在PG/PC上离线测试程序，可以使用所有的STEP 7编程语言（STL、LAD、FBD、S7－Graph、S7－HiGraph、S7－SCL和CFC）。

仿真PLC主要应用在以下场合：无硬件PLC时的调试程序；有硬件PLC，但在实际调

试时由于情况复杂可能会损坏 PLC 等。利用 S7－PLCSIM，可以在开发阶段就发现和排除错误，从而提高用户程序的质量并降低试车的费用。S7－PLCSIM 也是初学者学习 S7－300/400 编程以及程序调试的便捷工具。

## 5.9.2 S7－PLCSIM 的使用方法

S7－PLCSIM 提供了一个简便的操作界面，我们可以监视或者修改程序中的参数，比如直接进行对数字量的输入操作。当 PLC 程序在仿真 PLC 上运行时，我们可以继续使用 STEP 7 软件中的各种功能，比如在变量表中进行监视或者修改变量。S7－PLCSIM 的使用步骤如下：

### 1. 打开 S7－PLCSIM

可以通过 SIMATIC 管理器中工具栏上的按钮打开/关闭仿真功能。按下仿真按钮，打开 S7－PLCSIM 软件，如图 5-31 所示，此时系统自动装载仿真的 CPU。当 S7－PLCSIM 在运行时，所有的操作（如下载程序）都会自动与仿真 CPU 相关联。

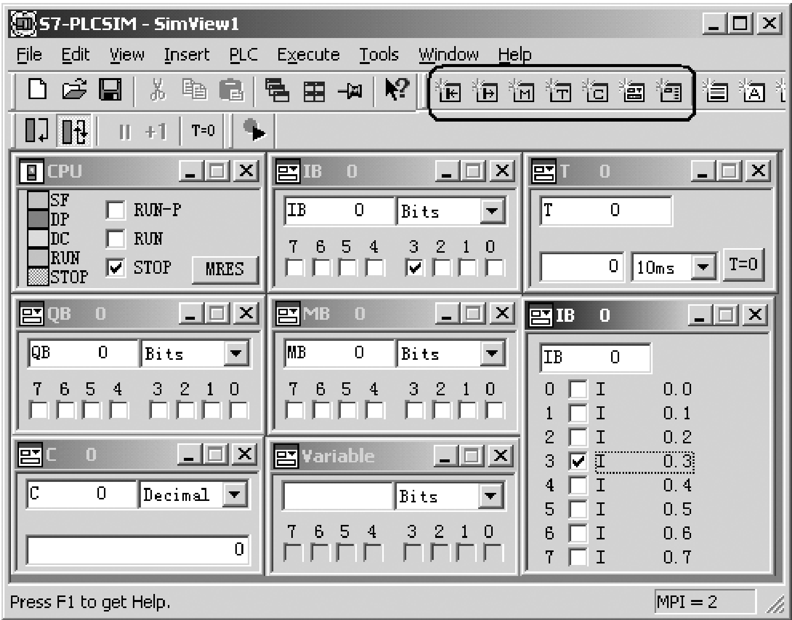



图 5-31 S7－PLCSIM 软件的界面

### 2. 插入“View Object”（视图对象）

通过生成观察对象（View Objects），可以访问存储区、累加器和被仿真 CPU 的配置。在观察对象上可以强制和显示所有数据。执行菜单命令“Insert/…”或直接点击如图 5-31 所示工具栏中的相应按钮，可以在 PLCSIM 窗口中插入以下视图对象：

① Input Variable：允许访问输入（I）存储区。

② Output Variable：允许访问输出（Q）存储区。

③ Bit Memory：允许访问位存储区（M）中的数据。

④ Timer：允许访问程序中用到的定时器。

⑤ Counter：允许访问程序中用到的计数器。

⑥ Generic：允许访问仿真 CPU 中所有的存储区，包括程序使用到的数据块（DB）。

⑦ Vertical Bits：允许通过符号地址或者绝对地址来监视或者修改数据。可以用来显示外部输入输出变量（PI/PO）、输入输出映像区变量（I/O）、位存储区、数据块等。

对于插入的视图对象，可以输入需要仿真的变量地址，而且可以根据被监视变量的情况选择显示格式：Bits、Binary、Hex、Decimal 和 Slider：Dec（滑动条控制功能）等。变量显示“Slider：Dec”的视图如图 5-32 所示，可以用滑动条的控制仿真逐渐变化的值或者在一定范围内变化的值。有三个存储区的仿真可以使用这个功能：Input Variable、Output Variable、Bit Memory。




图 5-32　变量显示“Slider：Dec”

### 3. 下载项目到 S7 - PLCSIM

在下载前，首先通过执行菜单命令“PLC/Power On”为仿真 PLC 上电（一般默认选项是上电），通过菜单命令“PLC/MPI Address…”设置与项目中相同的 MPI 地址（一般默认 MPI 地址为 2），然后在 STEP 7 软件中点击按钮将已经编译好的项目下载到 S7 - PLCSIM。若点击 CPU 视图中的 MRES 按钮，可以清除 PLCSIM 中的内容，此时如果需要调试程序，必须重新下载程序。

### 4. 选择 CPU 运行的方式

执行菜单命令“Execute/Scan Mode/Single scan”或单击工具栏中按钮，使仿真 CPU 仅执行程序一个扫描周期，然后等待开始下一次扫描；执行菜单命令“Execute/Scan Mode/ Continuous scans”或单击工具栏中按钮，仿真 CPU 将会与真实 PLC 一样连续地周期性的执行程序。如果用户对定时器 Timer 或计数器 Counter 进行仿真，那么这个功能非常有用。

### 5. 调试程序

用各个视图对象中的变量模拟实际 PLC 的输入/输出信号，用它来产生输入信号，并观察输出信号和其他存储区中内容的变化情况。模拟输入信号的方法是：用鼠标单击图 5-31 中 IB0 的第 3 位（即 I0.3）处的单选框中，则在框中出现符号“√”表示 I0.3 为 ON，若再单击这个位置，那么“√”消失，表示 I0.3 为 OFF。在“view object”中所作的改变会立即引起存储区地址中内容发生相应变化，仿真 CPU 并不等待扫描开始或者结束后才更新变换了的数据。执行用户程序过程中，可以检查并离线修改程序，保存后再下载，之后继续调试。

### 6. 保存文件

退出仿真软件时，可以保存仿真时生成的 LAY 文件及 PLC 文件，便于下次仿真这个项目时可以直接使用本次的各种设置。LAY 文件用于保存仿真时各视图对象的信息，例如选择的数据格式等；PLC 文件用于保存仿真运行时设置的数据和动作等，包括程序、硬件组态、设置的运行模式等。

### 5.9.3 S7 – PLCSIM 的调试应用举例

以电动机的星形 – 三角形（Y – △）起动的控制程序（见图 5–33）为例，说明用 S7 – PLCSIM 进行仿真的调试方法。

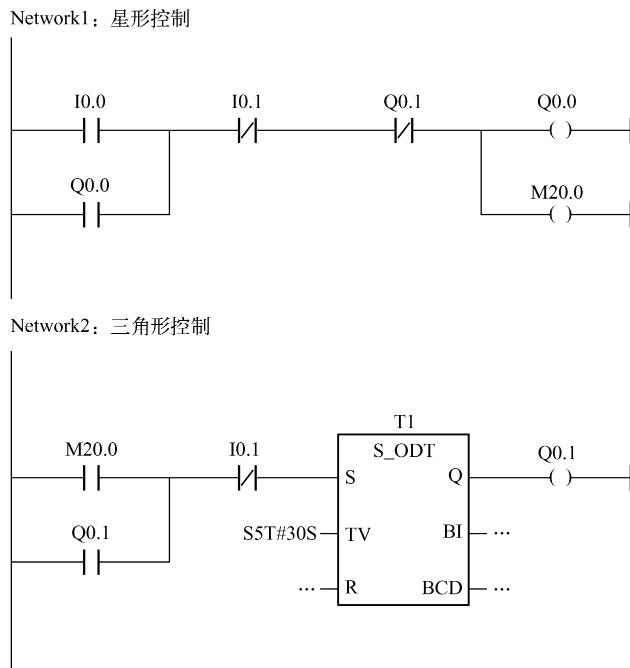



图 5–33 星形 – 三角形（Y – △）起动的控制程序

控制要求：按起动键 I0.0，电动机以星形方式起动 30s 后，以三角形方式全压运行；按停止键 I0.1 后，无论电动机处于何种方式运行，都立即停止运行。

说明：Network1 中，I0.0 得电使星形线圈 Q0.0 得电并自锁。Network2 中，利用 M20.0 触发 T1 起动，当定时时间到后，三角形线圈 Q0.1 得电，同时星形线圈 Q0.0 失电，使电动机全压运行。

在 STEP 7 中保存程序并下载到 S7 – PLCSIM 中，将 S7 – PLCSIM 中 CPU 的操作模式置于 RUN 或 RUN – P 状态。在 S7 – PLCSIM 中插入输入字节 IB0、输出字节 QB0 和定时器 T0。

用鼠标单击 I0.0 的单选框，在框中出现符号“√”，此时观察到 Q0.0 的单选框出现符号“√”，说明电动机以星形方式起动，再单击 I0.0 的单选框释放这个信号（相当于起动按钮是点动按钮）。在起动电动机的同时，定时器 T1 开始计时，30s 后电动机自动切换至三角形运行方式。

模拟定时器时间的方法：直接单击如图 5–34 所示 S7 – PLCSIM 调试界面中的“T = 0”按钮，可迅速达到计时时间。




图 5-34　S7 - PLCSIM 调试界面

## 5.9.4　仿真 PLC 与真实 PLC 的区别

### 1. 仿真 PLC 特有的功能

① 在 S7 - PLCSIM 中可人为地触发中断。主要包括 OB40 ~ OB47（硬件中断）、OB70（I/O 冗余错误）、OB72（CPU 冗余错误）、OB73（通信冗余错误）、OB82（诊断中断）以及 OB83（插入/移除模块）等。但不支持功能模块 FMs。

② 可以选择让定时器自动运行或者人为地进行置位/复位。我们可以针对各个定时器单独复位，也可以同时复位所有定时器。

③ 可以把仿真 CPU 当作真实的 CPU 那样改变它的运行模式（STOP/RUN/RUN - P）。此外 S7 - PLCSIM 提供“暂停”功能，允许暂时把 CPU 挂起而不影响程序的状态输出。

④ 可以记录一系列事件（复制输入输出存储区、位存储区、定时器、计数器），并能重放记录，实现程序测试的自动化。

⑤ 可以选择单次扫描或连续扫描。

### 2. 仿真 PLC 与实际 PLC 的区别

① PLCSIM 不支持写到诊断缓冲区的错误报文，例如不能对电池失电和 EEPROM 故障进行仿真，但是可以对大多数 I/O 错误和程序错误进行仿真。

② 不支持功能模块和点对点通信。

③ S7 - 300 的大多数 CPU 的 I/O 是自动组态的，模块出入物理控制器后被 CPU 自动识别。仿真 PLC 没有这种自动识别功能。如果将自动识别 I/O 的 S7 - 300CPU 的程序下载到仿真 PLC，系统数据没有包括 I/O 组态。因此在用 PLCSIM 仿真 S7 - 300 程序时，如果想定义 CPU 支持的模块，首先必须下载硬件组态。

④ 在视图对象中的变动会立即使对应的存储区中的内容发生相应的改变。实际的 CPU 要等到扫描结束时才会修改存储区。

总之，利用仿真 PLC 可以基本达到调试程序的目的。

通过本章节的介绍，可以深入体会“工欲善其事，必先利其器”的道理。

## 习题

1. 利用断点调试方法，试调试第 4 章例 12。
2. 利用参考数据调试如图 5-35 所示程序。

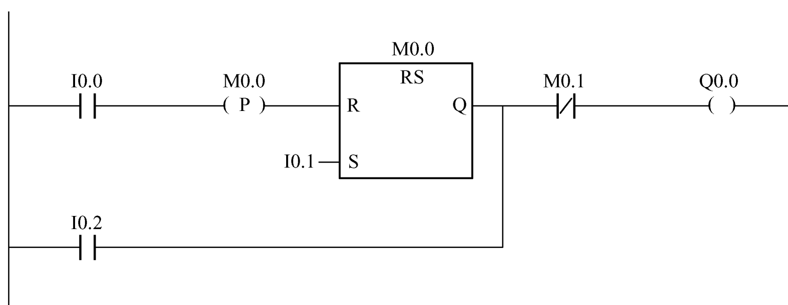



图 5-35　习题 2 程序

# 第6章　编程技术

从工业控制角度而言，控制系统的设计包含许多内容和步骤，设计者不仅需要具有丰富的专业知识，而且更需要有效的编程思路及方法，这样对系统的设计及调试、系统的扩展、合作者之间的协作交流都十分有利。

设计过程对自动化工程师来说是富于挑战的一项任务。原因有两个方面，其一是控制系统的特点：多样性（有许多不同的技术、元件，不同的工程师有不同的设计方案）；复杂性；单一性（有些系统不是批量生产）。其二是客户的要求：技术性（具有许多不同的功能）；经济性；质量保障（可靠性、便捷性、安全性——符合工业标准）。

控制系统的设计包括以下几个步骤：

1）分析任务（将任务分解为若干子任务）。

2）描述任务和子任务（任务说明书）。

3）生成控制算法。

4）选择控制媒介。

5）执行（建立控制系统）。

6）验证（测试整个系统）。

设计过程如图6-1所示。

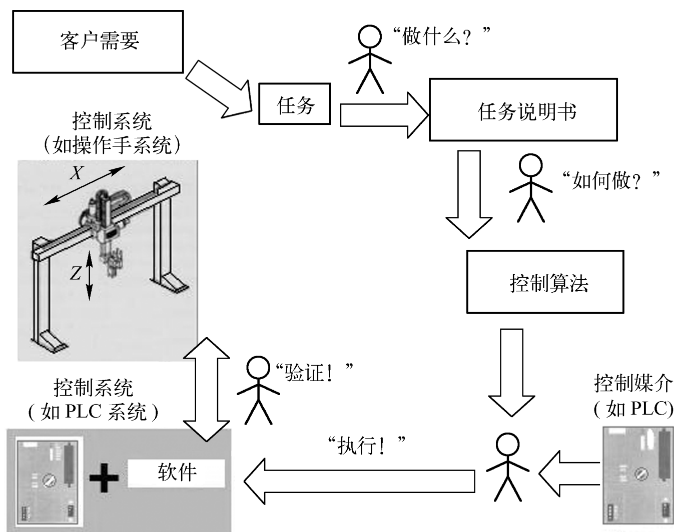



图6-1　设计过程

要完成控制系统设计任务，自动化工程师必须具备大量的知识、技能、方法和策略。其中需要的知识和技能有：有关控制系统的知识（系统性能、传感器、执行器）；有关控

制媒介的知识（PLC 等）；分析控制任务的技能；描述任务的技能；执行和验证的技能（PLC 编程技术和验证工具的使用）。需要具备的方法和策略包括：策划和组织工程的方法；描述控制任务的方法；系统化设计的策略（根据任务书要求，完成控制算法，生成 PLC 程序）；满足客户要求的技术解决手段（错误诊断和监控、缩短设计进程、简化软件结构）。

在可编程控制技术中，编程技术一般包括：经验法、功能图法、流程图法、解析法、顺序功能图法和状态图法。本书将着重介绍顺序功能图法和状态图法。

## 6.1 控制系统的基本设计步骤

上述的 6 个步骤简化为以下的 4 个步骤，分别加以说明。

### 6.1.1 分析和描述任务

在进行控制系统设计之前，自动化工程师首先应根据工艺流程定义控制任务和确定工作步骤。控制任务的定义为系统的设计奠定了基础，其定义必须由熟悉工艺流程的工程师和设计工程师联合进行，这样才能最大限度地减小由于错误理解工艺流程而产生的错误，同时也符合设计需要。

在大型工业控制过程的控制任务定义中，各部门的人员应商定需要多少个控制对象和信号采集点，以便每个人都知道在项目中要完成的任务。例如，在一个包含工厂制造自动化的项目中，需要从仓库中取物料并送至自动化包装区。在系统定义期间，仓库和包装区的工作人员都应与设计组协作，以便满足某些数据报告的需要以及管理方面的需要。

定义任务要考虑的因素与设计结果的成败息息相关。定义任务之后，为联合设计的需要，应将控制任务划分为若干子任务（或任务块），并确定相应的输入、输出接口。子任务划分的合理性会使系统的设计事半功倍。

### 6.1.2 确定控制策略

定义了控制任务之后，即可以开始规划其解决方案了。程序开发的这部分称之为算法开发。

算法对我们来说并不新奇，我们都是按算法来完成某些任务的。比如一个人从家去上学或去工厂的过程就可以看作是一个算法。如果要了解当今流行的复杂算法，各大科技杂志涉及的有小波变换、神经元、遗传算法等。好的算法能加快系统的运行速度，在实时性要求比较高的场合，尤为重要。在 PLC 的编程中，同样存在算法问题，有了好的算法、好的编程思路，程序开发就变得简单了。

### 6.1.3 决定运行方式

可编程序控制器控制系统有三种运行方式：自动、半自动和手动。自动运行方式是控制

系统的主要运行方式，只要运行条件具备，由操作人员确定并按下起动按钮后，控制器起动系统并自动运行。

半自动运行方式，即系统的起动或运行过程中的某些步骤需要人工干预方可进行。半自动方式多半用于检测手段不完善而需要人工判断或某些设备不具备自控条件而需要人工干涉的场合。

手动运行方式不是控制系统的主要运行方式，而是用于设备调试、系统调整和紧急情况下的控制方式，因此它是自动运行方式的辅助方式。所谓紧急情况是指控制器在故障情况下运行。从这个意义上讲，手动方式又是自动运行方式的后备运行方式，所以手动运行方式的程序一般不能进入控制器。

在运行方式设计的同时，还必须考虑到停运方式设计。可编程序控制器的停运方式有正常停运（正常停车）、暂时停运（暂时停车）和紧急停运（紧急停车）三种。

正常停运由控制器的程序执行，当运行步骤执行完，且无需重新起动执行程序时，或控制器接收到操作人员停运指令后，控制器将按规定的停运步骤停运系统。

暂时停运方式有以下几种：

在程序控制方式下，暂时停运方式暂停执行当前程序，使所有输出都置成 OFF 状态，待暂停解除后将继续执行被暂停的程序。

另一种暂停运行方式是用暂停开关直接切断负荷电源，同时送给控制器相应输入信号，以停止执行程序。

还有一种暂停运行方式是把 CPU 模块上的 RUN 切换成 STOP。这种方法最简单，且程序保留暂停前的状态。

紧急停运方式是，当控制系统中某一设备出现异常情况或故障时，如不立即停止运行，系统将导致重大事故或有可能损坏设备，这时必须使用紧急停运按钮使所有设备立即停运。它是既没有联锁条件也没有延迟时间的停运方式。紧急停运时，所有设备都必须停运，且程序控制被解除，控制内容复位到原始状态。

### 6.1.4 控制系统的调试

调试是控制系统的最后一个设计步骤，在完成总体设计以及具体的硬件和软件系统设计之后，除了要分别对硬件系统和软件系统进行调试外，还必须进行联合调试和试运行。反复进行硬件系统和软件系统的修改和调整，直到整个控制系统全部投入正常工作，最终完成系统的设计。

需要特别指出的是，在确定控制方式之后和进行 I/O 地址分配之前，必须进行外围电路的设计；在确定存储器容量之后和选择 I/O 模块之前必须进行选择外设的工作；在选择 I/O 模块之后和进行控制回路设计之前，必须进行控制盘/框的设计。这些工作可在上述适当的场合穿插进行，但在整个硬件设计过程中，是必不可少的工作。

## 6.2 编程技术基础

可编程序控制器可以完成以下几种控制：

1）代替普通继电器的顺序控制。
2）进行步进控制和顺序控制。
3）进行定时器的时序控制。
4）进行计数器的计数控制。
5）按高速计数器值进行计数、测量和位置控制。
6）根据模拟信号的输入进行测量控制。
7）根据模拟输入/输出信号进行 PID 控制。
8）用位置控制模块进行高速、高精度的位置控制。
9）用数据指令进行数值处理、参数控制等。
在编制复杂的步进控制或顺序控制程序时，必须考虑几个问题：
1）起动和自锁条件（保持条件）。
2）约束条件（如位置约束）。
3）关断信号条件。
4）联锁信号条件。
在全面考虑这几个问题之后，就能够安全、正确地完成设计任务。

### 6.2.1　程序设计举例

对于初学者，由于经验不足，在编程时，往往理不清条件之间的逻辑关系，或者考虑欠周全，这样，在系统调试时，将浪费大量的人力物力。

如果在编程中充分考虑控制条件，理顺逻辑关系，在软硬件方面对约束、关断等条件进行综合考虑，再进行设计，成功率会比较高。

下面以液压滑台式自动攻螺机为例，说明如何设计控制系统。

如图 6-2 所示为自动攻螺机简图，其中 1SQ、2SQ 和 3SQ 是检测滑台运行位置的行程开关，4SQ、5SQ 是检测丝锥运行位置的行程开关，KT1 是丝锥运行到终点时的电动机能耗制动开关。滑台的运动是由 3 个电磁阀打开和关闭油路控制，丝锥的运动是由一台电动机进行正反转控制。初始位置为：滑台处于原位 1SQ，丝锥处于原位 4SQ 处。

当按下起动按钮后，第一个电磁阀打开，油压将滑台快速推进到 2SQ，此时第二个电磁阀打开，滑台变为慢速前进。到 3SQ 时，吸合电动机接触器，丝锥正转前进。到达终点 5SQ 后电动机被制动停止。之后丝锥反转并后退到 4SQ，并再次能耗制动电动机。此时第三个电磁阀打开，油压将滑台快速推回到原位。如图 6-3 所示是自动攻螺机工序图。

攻螺机变量表如表 6-1 所示。

**表 6-1　攻螺机变量表**

| PLC 输入地址 | 变量名 | PLC 输出地址 | 变量名 |
|---|---|---|---|
| I0.0（1SQ） | 滑台原位 | Q4.0 | 电磁阀 1 动作（滑台快进） |
| I0.1（2SQ） | 滑台变速 | Q4.1 | 电磁阀 2 动作（滑台慢进） |
| I0.2（3SQ） | 滑台终点 | Q4.2 | 电磁阀 3 动作（滑台快退） |

（续）

| PLC 输入地址 | 变量名 | PLC 输出地址 | 变量名 |
|---|---|---|---|
| I0.3（4SQ） | 丝锥原位 | Q4.3 | 接触器 1 动作（丝锥正转） |
| I0.4（5SQ） | 丝锥终点 | Q4.4 | 接触器 2 动作（丝锥反转） |
| I1.0 | 起动按钮 | | |

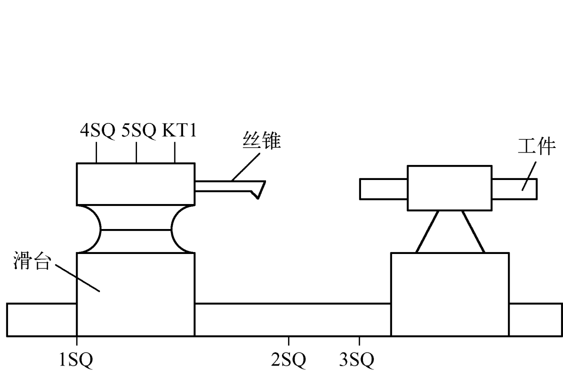



图 6-2　自动攻螺机简图

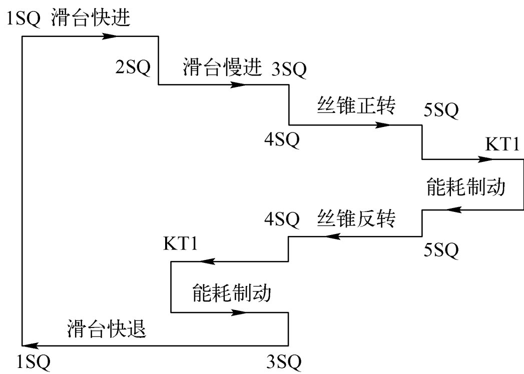



图 6-3　自动攻螺机工序图

在 STEP 7 中建立攻螺机系统的符号表如图 6-4 所示。

S7 Program(1) (Symbols) -- huatai\...

| | Status | Symbol | Address | Data type | Com |
|---|---|---|---|---|---|
| 1 | | 1SQ | I 0.0 | BOOL | |
| 2 | | 2SQ | I 0.1 | BOOL | |
| 3 | | 3SQ | I 0.2 | BOOL | |
| 4 | | 4SQ | I 0.3 | BOOL | |
| 5 | | 5SQ | I 0.4 | BOOL | |
| 6 | | 滑台快进 | Q 4.0 | BOOL | |
| 7 | | 滑台快退 | Q 4.2 | BOOL | |
| 8 | | 滑台慢进 | Q 4.1 | BOOL | |
| 9 | | 丝锥反转 | Q 4.4 | BOOL | |
| 10 | | 丝锥正转 | Q 4.3 | BOOL | |
| 11 | | 启动按钮 | I 1.0 | BOOL | |

图 6-4　攻螺机系统的符号表

自动攻螺机基本控制梯形图如图 6-5 所示。

说明：程序中 Network1 表示滑台快进，Network2 表示滑台慢进，Network3 表示丝锥正转，Network4 表示丝锥反转，Network5 表示滑台快退。并加入了关断条件。图 6-5 所示的程序似乎能满足控制要求，可实际上程序存在着很大的缺陷。如在第二段的滑台慢进中，如果丝锥在起动时不在原位，滑台运行到 3SQ 位置时，会造成丝锥的折断。另一方面，如若

在起动时，控制滑台快退的电磁阀没有关闭，滑台根本就无法动作。所以从安全的角度考虑，还必须在程序中加入约束条件和联锁信号条件。

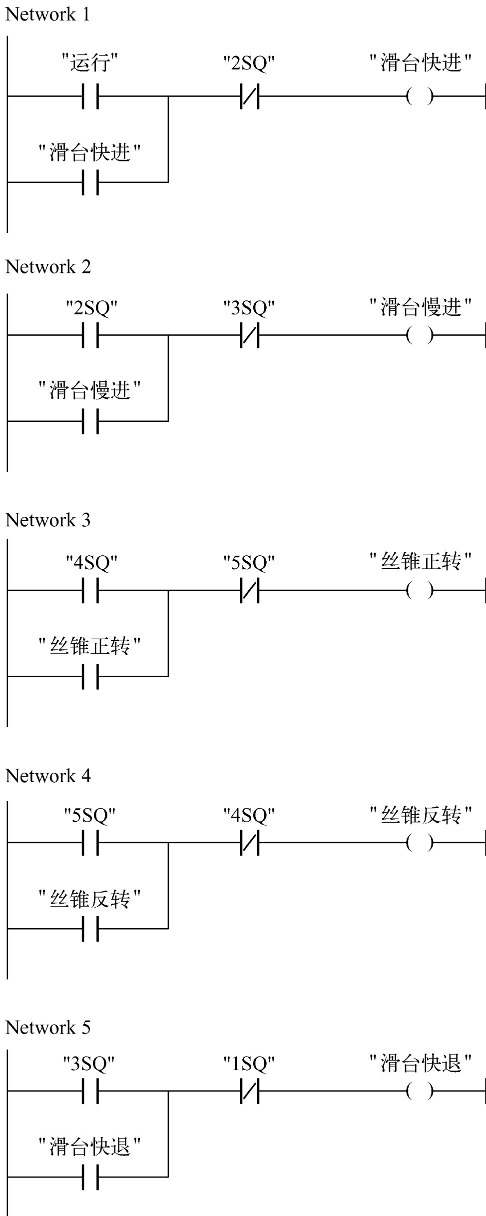



图 6-5　自动攻螺机基本控制梯形图

改进后液压式自动攻螺机梯形图如图 6-6 所示。

说明：在程序的 Network1 和 Network2 和 Network5 中加入了丝锥在原位的约束条件，同时加入滑台快退的互锁条件；在 Network3 和 Network4 中加入丝锥正、反转之间的互锁；在 Network5 中，加入滑台快进的互锁条件。

可见，加入恰当的约束和互锁条件后，系统的可靠性大大加强。

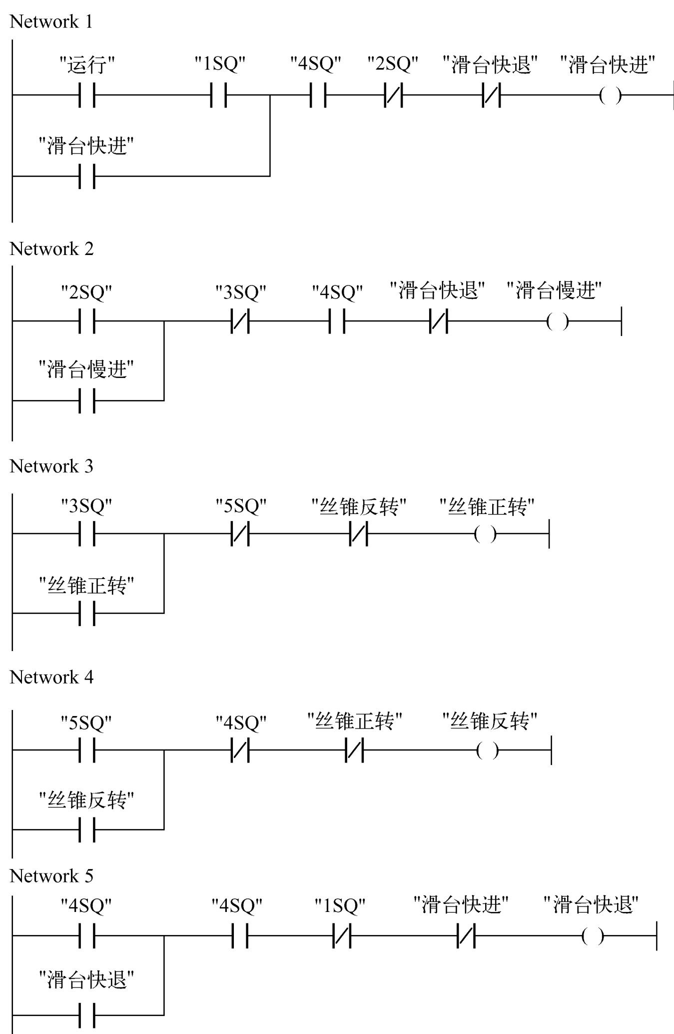



图 6-6　改进后液压式自动攻螺机梯形图

### 6.2.2　编程要求

编程的基本要求如下：

**1. 所编程序合乎 PLC 有关规定**

所谓合乎规定是指对指令理解准确并能正确使用。各种 PLC 指令大同，但存在小异，所以在使用时，注意不要“张冠李戴”。

**2. 所编程序尽可能简明**

程序简明则节约内存，简化调试，并且节省程序执行时间，提高对输入的响应速度。

**3. 所编程序便于交流**

因为在大系统设计中，需要分工合作，程序的规范化编程将使合作者便于交流。在编写

程序中同时还需注意程序的层次，讲究模块化、标准化；同时还需要建立相应的符号表。

**4. 所编程序合乎 PLC 性能指标及工作要求**

主要应注意 3 点：

1）程序的指令条数满足 PLC 的内存容量。

2）输入/输出点数要在所选用 PLC 的 I/O 点数范围内，并留有一定的余量。

3）PLC 的扫描时间要小于所选用 PLC 程序的运行监控时间（Watch dog time）。

其中 PLC 的扫描时间不仅包括运行用户程序所需要的时间，而且还包括运行系统程序（如自检测 I/O 处理等）所需时间。所以，扫描时间与系统的结构、I/O 模块的个数以及外设有关。

扫描时间影响输出对输入的响应时间。所以这个时间太长，将降低控制的实时性。

**5. 所编程序能够循环运行**

这是 PLC 顺序扫描控制的特点。运行从初始化后的状态开始，待控制对象完成工作循环后，则应返回初始化状态。只有这样，控制对象在新的工作周期中才能得到相同的控制。

## 6.3 控制系统分析方法及系统建模

### 6.3.1 控制系统分析方法

控制系统分析方法主要包括：经验法、功能图法、流程图法、解析法（布尔等式）和图解法（顺序功能图和状态图）。

经验法：运用自己或他人的经验进行设计。在前面的程序设计举例中，就是运用了这种方法。用经验法设计时，没有一套固定的方法和步骤可以遵循，具有很大的试探性，在设计复杂系统时，需用大量的中间单元来完成联锁、互锁等功能。由于需要考虑的因素较多，彼此互相影响，所以很难把所有问题考虑周全。即使有经验的工程师也很难做到一次设计成功，并且由经验法设计的程序很难阅读，给系统的维修和调试带来较大不便。

解析法：根据组合逻辑或时序逻辑，运用相应的解析方法，对其进行逻辑关系的求解，之后，根据求解结果编写程序。解析法比较严谨，用户可以运用一定的标准，使程序优化并算法化。解析法可以避免编程的盲目性，是比较有效的方法。但由于其抽象分析的特点，所以分析系统比较复杂。

功能图法是利用时序波形分析系统，而流程图法则利用基础流程图，两者可以同解析法联合进行系统的设计。

图解法：它是我们推崇的行之有效的设计方法。图解法包括顺序功能图法和状态图法，运用顺序功能图和状态图等编写的程序具有直观的特点，便于合作者之间的交流，便于系统的再次开发。

图解法的具体分析步骤如下：

1）了解 PLC 程序的执行过程及输入、输出关系。

2）根据相应图形的画法规则，画出顺序功能图或状态图。

3）编写相关的梯形图程序或 STL 程序。

上述几种编程方法的特点比较如表 6-2 所示。

**表 6-2　编程方法的特点比较**

| 编程方法 | 特点 |
|---|---|
| 经验法 | 有盲目性，不严谨，缺乏模块化和系统化分析 |
| 功能图<br>流程图<br>解析法（布尔等式） | 具有容易理解的任务描述，适合于系统的分析和设计，但分析方法综合应用的效果更好，因此增加了分析的难度，直观性不强 |
| 顺序功能图<br>状态图 | 系统模型可以直接转换成程序，具有编程定式，更加严谨，更加直观 |

## 6.3.2　系统建模

系统建模从广义上讲是指工程模型的建立。工程模型是一种活动模型，它可运用于许多场合，如机械工程、电力工程、软件工程等。工程模型描述了整个的工程过程。它是由若干条语句组成的，每一条语句与在不同领域的不同行为息息相关。表 6-3 中显示了一个设备制造公司的工程语句和与输入/输出数据相关的行为。

**表 6-3　工程语句和与输入/输出数据相关的行为**

| 输入（来自客户） | 工程语句 | 行为 | 输出（到客户） |
|---|---|---|---|
| 调查 | 分析 | 问题分析，需求分析 | 报价 |
| 订单 | 定义 | 定义需求，设计（电气/气动/…统计/软件） | 建立文档，任务描述，与客户间的通信服务 |
| 设计文档的正式批准 | 执行 | 实现，生产包括测试 | 产品 |
| 正式移交 | 安装 | 操作环境的建立，起动 | 实用设备 |
| 委托事项 | 操作 | 服务 | 客户受益 |

在这些工程语句中包含了控制工程师的任务，即在接收到任务的技术简图后，首先必须进行系统分析，对任务进行工程描述；其次进行硬件和软件方面设计；之后，实现和测试；最终，建立操作环境。

对任务的工程描述是其中比较重要的环节。要想生成一个“Bug Free”的任务描述，必须具备完整性、简洁性和无矛盾性。在许多场合下，口语描述一个任务常常不太准确，而且有效性不高，而补充的技术简图会减少人类语言的匮乏，使之更趋近于工程，但在许多场合下仍然不能满足自动化工程师的需要。因此必须要有行之有效的表示方法，前面我们已经提到了若干个分析方法：如功能图、流程图等。在建立控制系统工程模型时，一种或多种任务的描述将会更好地完成系统的建模工作。

可见，在工程模型中，任务描述只是为系统的设计提供必要的前提条件。

系统设计方法可以用两大部分来实现。

**1. 独立于技术的描述逻辑控制任务的方法**

这些方法提供完善的任务描述，为系统的执行打下基础。具体为：

1）功能图。

2）流程图。

3）布尔等式（解析法）。

4）状态图。

5）顺序功能图。

**2. 依据逻辑控制任务的描述，执行控制任务的手段**

1）梯形图。

2）功能块图。

3）GRAPH 语言。

### 6.3.3 工程实例

如图 6-7 所示为果汁加工浓缩示意图，表示了果汁加工的工艺过程。设计任务有：

1）用功能图和解析法描述系统。

2）用功能块图进行程序的实现。

3）分析可能进行的系统监控。

这里我们将通过此例介绍如何利用功能图和解析法将口语描述转变为有效的逻辑控制任务。功能图和解析法归属于同一类分析方法，其特点是严谨，但分析比较复杂。

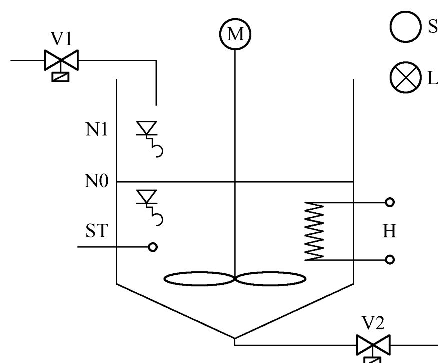



图 6-7　果汁加工浓缩示意图

**1. 用功能图和解析法描述系统**

（1） 口语描述

果汁在加热罐中的加热和浓缩过程如下：

1） 在指示灯 L 亮的前提下，按下 S 开关，进料阀门 V1 打开，果汁通过 V1 阀加入罐中，当果汁高度上升至 N1 高度处，阀门 V1 关闭。

2） 在果汁高度超过 N0 时，则电动机 M 工作，开始搅拌果汁。

3） 加热器 H 在果汁高度超过 N0 时也开始工作，加热的果汁温度保持在 70 ~ 80℃之间，由温度传感器 ST 检测温度值。

4） 当果汁加热，液面高度浓缩到低于 N0 时，浓缩过程结束。打开排空阀 V2 排出果汁。为完全排空果汁，阀门 V2 应持续打开 30s。

5） 当果汁排空，指示灯 L 亮，等待下一次的果汁浓缩过程。

其中液位高度传感器的感应高度分别为 N0 和 N1；N0 表示加工果汁的浓缩高度，N1 表示果汁装满果汁罐。

（2） 正规描述法

为了得到准确而又无矛盾的描述，同时能很好理解系统的功能，必须运用正规描述法（功能图法、流程图法、解析法、顺序功能图法和状态图法） 描述系统功能。对于工程师来讲，正规描述可以作为准确描述控制预期行为的模型。某些正规描述（如顺序功能图和状态图）甚至可以直接作为控制软件来执行，其表现形式为编程手段或编程语言。这些方法更直接，从而避免了设计到执行的过程。在 STEP 7 V5.3 版本中有 S7 Graph（顺序功能图）

编程软件。但 S7 High Graph（状态图）编程软件属自选软件包，用户必须额外购买。正规描述系统功能的另外一个原因是为了在某些场合能够更容易地理清系统的执行过程，同时还可以得到很有用的文档，便于系统的再设计和技术改造。本例将运用功能图和解析法进行控制系统的分析。

首先进行信号的逻辑赋值（0/1），以决定其活动状态。信号赋值表如表 6-4 所示。

**表 6-4　信号赋值表**

| 传感器 | 输入地址 | 功能/状态 | 信号赋值 |
|---|---|---|---|
| 起动按钮 S | I0.0 | 起动 | 1 |
| N0 高度指示 | I0.3 | 高度超过 | 1 |
| N1 高度指示 | I0.2 | 高度超过 | 1 |
| 温度传感器 ST | I0.4 | 温度超过 80℃ | 0 |
| 执行器 | 输出地址 | 功能/状态 | 信号赋值 |
| 电动机 M | Q0.2 | 搅拌/运行 | 1 |
| 灌入阀 V1 | Q0.1 | 加满果汁罐 | 1 |
| 排空阀 V2 | Q0.4 | 排空果汁罐 | 1 |
| 加热器 H | Q0.3 | 加热果汁 | 1 |
| 准备好指示灯 L | Q0.0 | 照明指示 | 1 |

使用信号赋值表必须考虑两个问题：

1）功能/状态列和信号赋值列必须与传感器/执行器的状态一致。

2）逻辑信号的赋值必须考虑信号断线的影响。

从信号表中可以看出系统有 5 个输出：阀门 V1、V2 的控制，电动机 M 的起动、加热器 H 的起动、指示灯 L 的控制。下面分别对输出进行分析，以便得到执行器的公式化描述。

如图 6-8 所示为果汁加热功能图，它可以更准确地描述加热过程。图中在果汁高度大于 N0 且温度 T 在 70～80℃之间时，经过数次加热器的开启和关闭过程，终于使果汁高度浓缩至小于 N0，加热过程并不是持续进行的，只有当温度 T 在一定的范围内才开始。

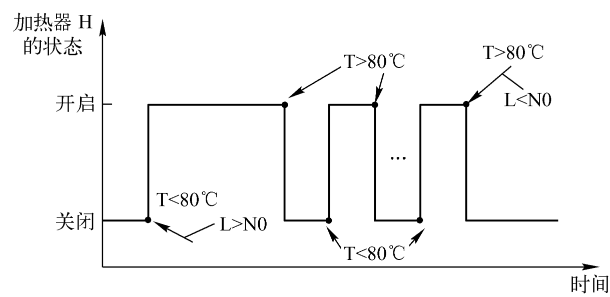


图 6-8　果汁加热功能图

加热器 H 的操作用解析式可以表示为：

$$H = N0 \cdot ST$$

其含义表示当果汁高度超过 N0，且温度传感器感应温度低于 80℃时，两个传感器的信号状态为“1”，此时加热器 H 开始工作。

电动机 M 的起动是在果汁高度超过 N0 时开始的，此时 N0 传感器的信号状态为“1”，

所以电动机 M 的操作用解析式可以表示为：

$$M = N0$$

在口语描述中，如果果汁罐未空，或者浓缩过程正在进行时，我们并不能决定何时能打开开关 S，并灌入果汁。这意味着可能会在浓缩的果汁中，掺入新鲜的果汁。这是口语描述存在问题的典型例子。此外，在口语描述中还可能出现一个问题被描述了两次，且两次的描述又互相矛盾的情形。为避免上述不准确的任务描述，我们使用了正规的表达方式：解析式。

果汁罐的加入操作用解析式可以表示为：

$$V1 = S \cdot L \cdot \overline{N1} + V1 \cdot \overline{N1}$$

其中解析式的第一项表示 V1 起动的条件，第二项为 V1 的自保持条件。

很明显，在加热过程中，不能允许新鲜果汁的灌入。这一点在公式中有了具体表现：在 N1 高度没有被超过，指示灯 L 亮（加热结束）的前提下，S 按钮被按下时，阀 V1 被打开。由于指示灯亮，从而保证了新鲜果汁不会在加热过程中灌入。这个描述到目前为止已经比较准确了。

如图 6-9 所示为果汁加工过程的功能图。图中表示了指示灯 L、阀门 V1 和 V2 以及高度 N0 之间的状态关系。图中可以看出在 V1 有效、V2 有效或者在果汁高度没有低于 N0 的情况下，L 是无效的。所以指示灯 L 的解析式可以表示为：

$$L = V1 + N0 + V2$$

根据摩根定律，将等式化简为：

$$L = \overline{V1} \cdot \overline{V2} \cdot \overline{N0}$$

当果汁高度浓缩到低于 N0 时，触发扩展脉冲定时器 T1，在 T1 工作期间，果汁排出，排空阀 V2 打开。其解析表达式为：

$$V2 = T1$$

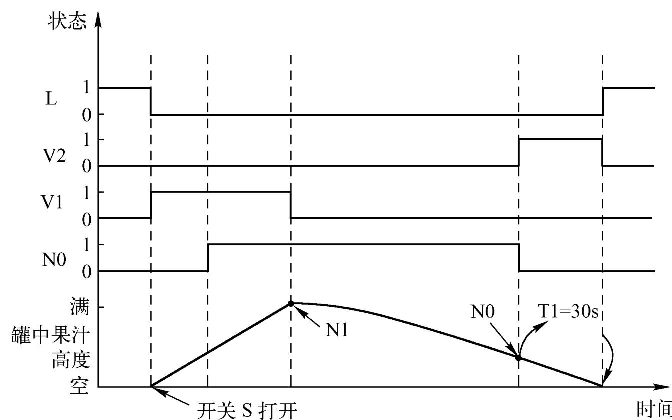
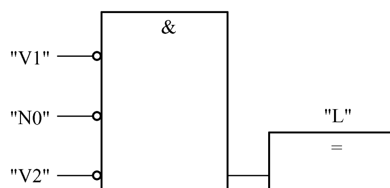



图 6-9　果汁加工过程的功能图

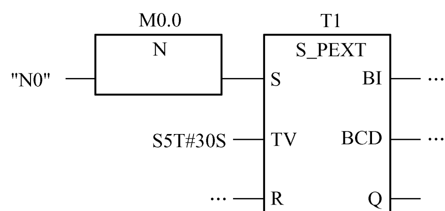
### 2. 用功能块图进行程序的实现

根据解析表达式，得出果汁加工系统功能块图程序，如图 6-10 所示。

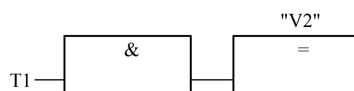
Network 1



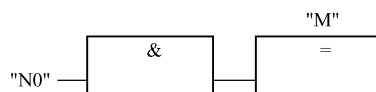

Network 2



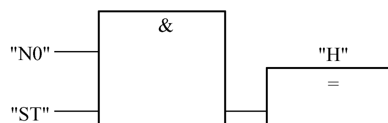

Network 3




Network 4




Network 5




Network 6

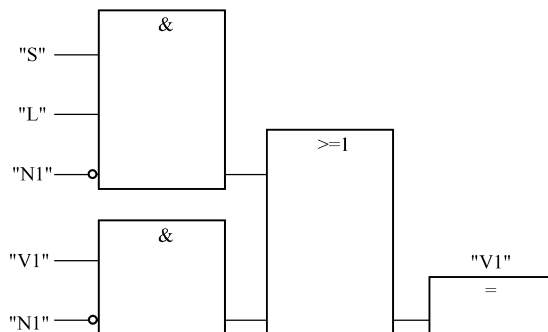



图 6-10　果汁加工系统功能块图程序

说明：在程序的 Network2 中通过 N0 的下跳沿触发扩展脉冲定时器；在 Network3 中，实现定时器 T1 工作期间，排空阀门 V2 打开。其余的段落的功能与解析表达式一致。

3. 分析可能进行的系统监控

如果灌入果汁高度总是高于 N1 或低于 N0 时，系统出现故障。出错的可能是由于测量高度的传感器的电线断了。监控的方法是将信号$\overline{N0}$和 N1 相与后置位错误指示灯。

如果阀 V1 打开时间过长，出错的可能是没有果汁；阀 V1 损坏或者测高传感器出错。

监控的方法是用 V1 起动接通延时定时器，定时时间到则错误指示灯亮。

如果温度长时间低于 80℃，出错的可能是加热器出故障，方法同上，起动定时器来点亮错误指示灯等。

从工程实例中可以看出系统的设计运用了功能图法和解析法。其中解析法将控制模型公式化，从而避免了编程的盲目性。但是对于复杂的大型系统，将很难从大量的功能图的描述中得出解析式，所以在进行系统分析时，功能图法和解析法并不是行之有效的设计方法。而顺序功能图法和状态图法具有先进的分析思想。相信用户在以后的系统分析中会逐步发现这两种设计方法的优点。

## 6.4 顺序功能图（SFC）

### 6.4.1 概述

1. 顺序功能图的历史

顺序功能图（Sequential Function Charts）最初是在 20 世纪 70 年代根据 DIN 19237 标准来制定的，同时，还有一个法国标准，称为 GRAPHCET。而在 1993 年由 IEC（6）1131 再次为 SFC 制定了新的标准。三个标准之间有一些小的区别，最初的标准中 SFC 不允许在控制中有分支或者有闭环，而在 IEC（6）1131 标准中提供了这种可能性。在本节中，我们并不介绍这种编程软件的用法，只是研究其编程思想。

2. 顺序设计法

顺序控制是指按照生产工艺预先规定的顺序，在各种输入信号的作用下，根据时间顺序，执行机构自动并有序地进行操作。顺序控制在各种生产流水线上应用非常广泛。

顺序控制设计是一种先进的设计方法，其设计思想是将系统的工作周期划分为若干顺序相连的阶段，我们称之为“步”。当步被激活时（即满足一定的转换条件），步所代表的行动或命令将被执行。这样一步一步按照顺序，执行机构就能够顺序“前进”。就像玩飞行棋一样，当扔出点数（满足转换条件）后，棋兵可以从当前格前进到下一个对应的格。运用顺序控制法能够提高设计效率，并且便于合作者进行编程思想的交流，同时程序的调试和修改也将十分方便。

设计步骤如下：

1）首先根据工艺流程，画出顺序功能图（SFC）。

2）翻译成梯形图（或功能块图）程序。

## 6.4.2　顺序功能图的绘制方法

### 1. 绘制基础

顺序功能图体现的是顺序控制的详细过程。其基本元素是由“步”组成的，每一步都包含三个要素：当前步、转换条件以及步的动作。

如图 6-11 所示为 SFC 步的表现形式。由图中可以看出所谓步即系统当前所处的状态，我们可以称之为“当前步”；转换条件是前一步进入当前步所需要的条件信号。转换条件可以是外部的输入信号，如按钮、开关等，也可以是 PLC 内部产生的信号，如定时器等提供的信号，当然转换信号也可以是这些信号的逻辑组合。步的动作是指当前步所执行的具体命令。图中还有辅助元素：前一步和后一步。这一点很好理解，没有前一步，怎么能有当前步？没有当前步，又怎么能有后一步。这样各个步之间形成一个链条，实现系统的顺序控制。由于 PLC 循环扫描，所以此链条封闭形成一个回环。

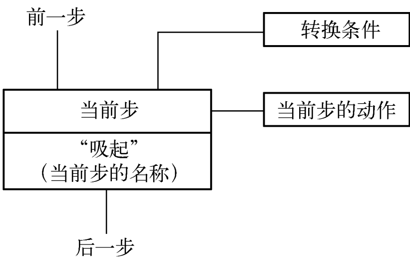


图 6-11　SFC 中步的表现形式

步的动作类型有几种形式，如表 6-5 所示为步的基本动作类型。在对应的动作中有存储型（S）和非存储型（NS）两大常用类型。存储型为保持型，可以用 S 和 R 指令对存储型动作置位和复位。而非存储型则与它所在的步“同存亡”，用输出线圈指令实现。

**表 6-5　动作类型**

| 命令类型 | 说明 | 命令类型 | 说明 |
|---|---|---|---|
| S | 存储命令 | ST | 存储并限时命令 |
| NS | 非存储命令 | D | 延迟命令 |
| SH | 存储，在电源故障时 | SD | 存储并延时命令 |
| T | 限时命令 | NSD | 不存储，延时命令 |

下面以图 6-12 说明步的动作。即在当前步进行 3 个动作：点亮指示灯（保持型）、电动机运行（非保持型）同时提升汽缸下降（保持型）。图中包括命令类型、动作的文字描述以及命令序号。

下面以具体的例子说明顺序功能图的画法。

**例 1**　如图 6-13 所示为彩灯循环点亮示意图。设五个彩灯的输出分别为 Q0.0、Q0.1、Q0.2、Q0.3 和 Q0.4，图中 I0.0 为控制开关。当 I0.0 打开时，彩灯依次顺序点亮（当一盏灯亮时，前一盏灯灭），点亮的周期为 2s。试画出系统的顺序功能图。

分析：在按下起动按钮 I0.0 后，彩灯系统开始工作，其工作周期包括：2s 时间到，第一盏灯亮；2s 时间到，第二盏灯亮（第一盏灯灭）；2s 时间到，第三盏灯亮（第二盏灯灭）；…2s 时间到，第五盏灯亮（第四盏灯灭）五个过程。

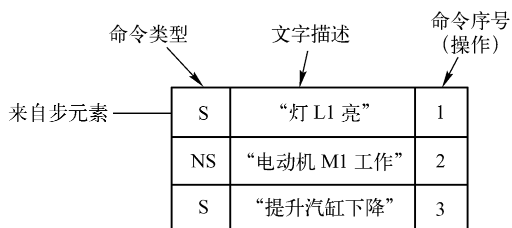



图 6-12　步的基本动作描述

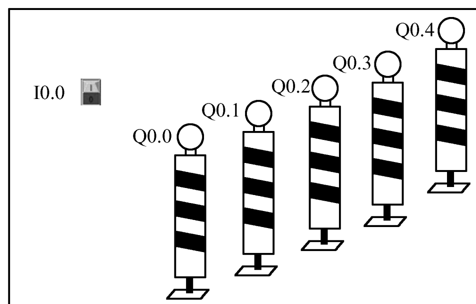



图 6-13　彩灯循环点亮示意图

如图 6-14 所示为彩灯系统的顺序功能图。

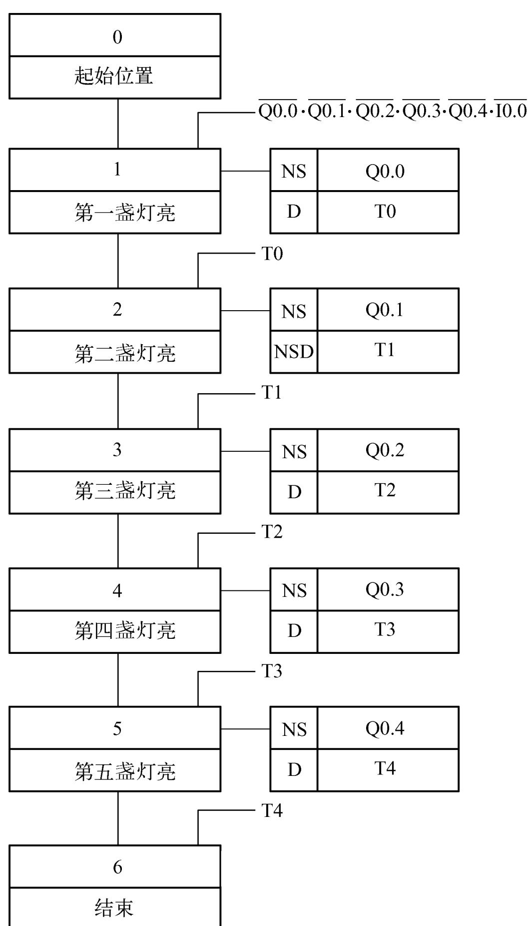



图 6-14　彩灯系统的顺序功能图

说明：经分析，系统工作需要五个步骤，加上起始和结束步，相当于有 7 步。起始位置状态（初始条件）是所有灯都灭，此时按下起动按钮 I0.0，满足转换条件，则系统从第 0 步转换到第 1 步，第 1 步的动作为点亮第一盏彩灯 Q0.0，并延时 T0 秒；当延时时间 T0 到，

系统从第 1 步转换到第 2 步，步的动作为点亮第二盏灯 Q0.1，并延时 T1 秒。以此类推，直至第五盏彩灯亮，延时 T4 秒，当延时时间 T4 到时，步结束，或者准确地说再次回到第 1 步，使第一盏灯亮。图中的每一步都包含着步的三要素：当前步的状态，转换的条件及步的动作，而每一步的前后又分别存在其他的步，以此达到了顺序控制的目的。

**2. 顺序功能图的基本结构**

顺序功能图的基本结构包括：单序列、选择序列、并行序列几种形式，如图 6-15 所示为简明图，着重强调步之间的跳转关系。

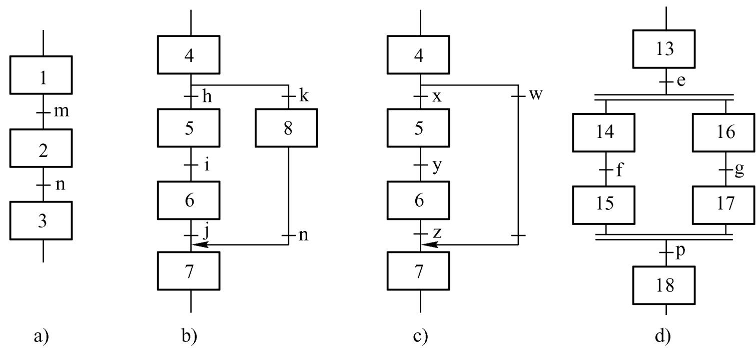



图 6-15　单序列、选择序列、并行序列形式

（1） 单序列

单序列是由顺序激活的步组成的，其特点是：没有分支与合并（见图 6-15 a）。

（2） 选择序列（多分支序列）

选择序列（见图 6-15 b、c）有分支，且转换条件要写在分支线以内。当转换条件 h = 1 时，功能图由第 4 步转为第 5 步。当转换条件 k = 1 时，功能图由第 4 步转为第 8 步。选择序列的结束称为合并。转换条件必须在合并线以内。图 6-15c 中显示的是跳步，即选择序列的某一条分支上没有步。但转换条件仍然存在。

（3） 并行序列

并行序列（见图 6-15d）是某一转换条件实现几个序列的同时激活。如图在转换条件 e = 1 时，14 和 16 步同时变为活动步。在表示同步的双水平线上，只允许有一个转换信号。并行序列表示系统的几个独立部分同时工作的情况。并行序列的结束同样称之为合并。在表示同步的双水平线之下，只允许有一个转换条件，即当转换条件 p = 1，且 15 步和 17 步同时为当前步时，会发生第 15、17 步到第 18 步的转变。此时，第 18 步变为当前步，而 15、17 步同时变为前一步。

**3. 绘制方法及绘制原则**

从简明图 6 - 15 可以看出，只要能说明问题，顺序功能图可以绘制得非常简单。但是简明图里不包含动作，会带来编程的麻烦，所以这里改进一下，形成顺序功能图的简易画法。在简易画法中，每一步用存储位来代表，比如 M0.0、M0.1 等；转换条件用一小横杠表示，旁边标注转换条件；选择序列的分支方向由带箭头直线表示；并行序列的分支和合并用两条平行线表示。当前步的动作前如果有字母 S 表示保持型动作的置位，有字母 R 表示保持型的复位，什么都没有表示非保持型动作。字母 D 表示延时动作。

这样，例 1 中的彩灯系统可以用简易画法来实现，如图 6-16 所示。

说明：初始步用存储位 M0.0 表示，其余的用递增的存储位表示。与图 6-14 的区别在于，没有了结束步，因为循环控制的要求，当最后一盏灯亮后，紧接着是第 1 盏灯亮。所以以选择序列合并的形式完成了循环控制的要求，返回 M0.1 步。步的动作包括了两个方面，第一是彩灯亮，由于灯亮是非保持型动作，所以没有表示，而时间为延迟时间，所以以 D 表示。

注意，如果最后一步完成后，直接返回到初始步，则理解为单周期运行方式。因为开始按钮按下的动作是瞬间动作，当运行完一周期之后，由于开始按钮为假，所以转换条件不满足，因此不能进行第二周循环。

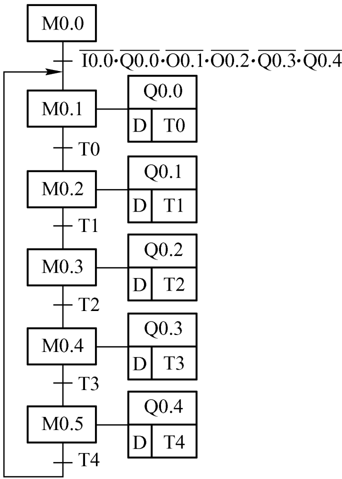



图 6-16　彩灯系统的简易顺序功能图

通过上述例子，可以总结归纳出顺序功能图的绘制要点。在绘制顺序功能图时，需注意：

1）两个步不能直接相连，必须用一个转换将两者隔开（如果没有具体的转换条件，一般都用定时延时来解决）。

2）两个转换条件也不能直接相连，必须用一个步将两者隔开。

3）起始步（起始位置）十分重要，它是进入顺序控制环（循环扫描）的入口，必不可少。

4）起始步如何变为当前步呢？有几种方法：在 PLCSIM 仿真调试时，将起始存储位打勾即可。在真实 PLC 调试时，可以用变量表将起始存储位改变为真。而在真正运行时，在 OB100 中将起始步预置为当前步，否则，系统不能正常工作。如果有手动、自动两种工作方式，在手动进入自动工作方式时，需用适当的信号将起始步变为当前步。

5）在单周期和循环自动运行方式下，结束步返回的方式不同。单周期运行方式下，返回起始步；循环运行方式下，返回第 1 步。

每个系统的顺序功能图的绘制不是唯一的，由于个人思想的不同，顺序功能图也呈现出多样化的特点。

下面以十字路口交通灯控制系统进行说明。

**例 2**　如图 6-17 为十字路口交通灯系统。设计要求为：开始按钮 I0.0 按下后，南北方向红灯亮 8 秒后，跳至绿灯亮 5 秒，之后黄灯亮 3 秒，如此循环。与之对应的是东西方向的绿灯、黄灯和红灯依次亮。设南北方向红灯为 Q0.0、绿灯 Q0.1 和黄灯 Q0.2，东西方向红灯为 Q1.0、绿灯 Q1.1 和黄灯 Q1.2。试绘制顺序功能图。

分析 1：如上所述，南北方向红、绿与黄灯依次亮时，东西方向则为绿、黄和红灯亮，且南北和东西方向同时交替闪亮，同时进行，所以可以用并行序列表示，如图 6-18a 所示。

分析 2：我们知道当南北方向红灯亮时，正好为东西方向绿灯与黄灯亮，同样，当东西方向转为红灯亮时，正好为南北方向绿灯和黄灯亮，如果暂时不考虑两个方向的红灯，则东西方向绿灯亮过，是黄灯亮，之后为南北方向的绿灯，之后为南北的黄灯，如此循环，这样，确定系统的主干步为四步，之后将两个方向的红灯在相应的步中体现即可。如图 6-18b 所示为另一种十字路口交通灯的顺序功能图。

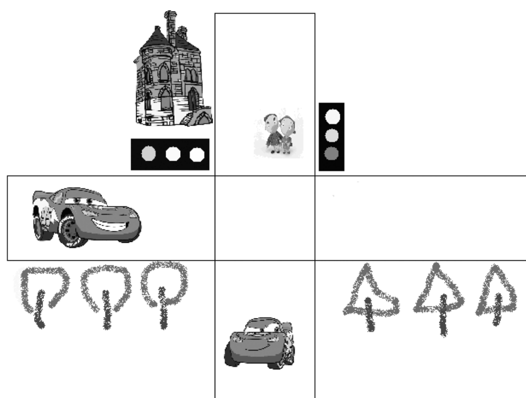


图 6-17　十字路口交通灯系统

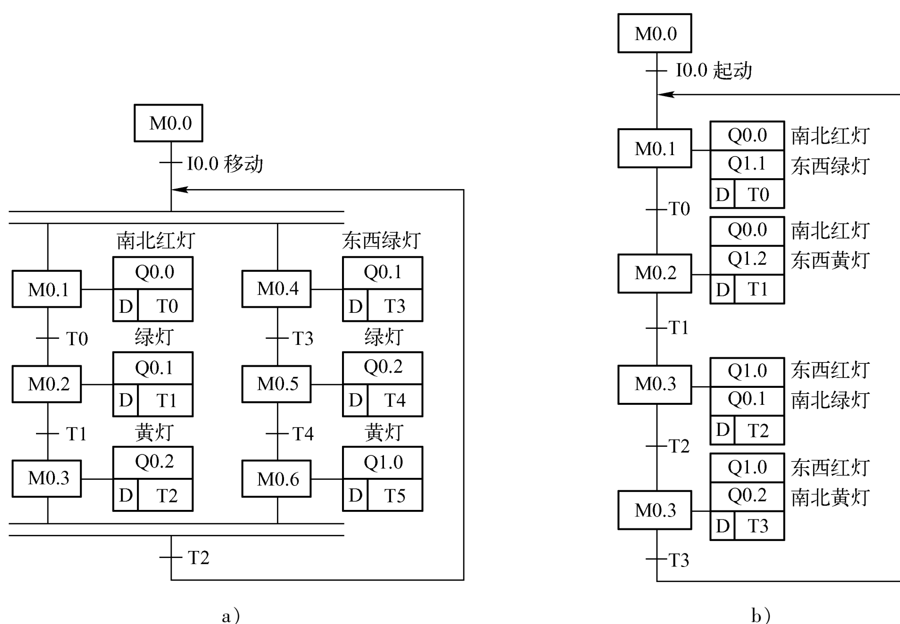



图 6-18　十字路口交通灯顺序功能图

虽然顺序功能图表现形式可以多样化，但是同一个顺序功能图，如果用相同的翻译方式，其程序却是基本一致的（排除掉程序的先后顺序）。下面介绍程序的编写。

### 6.4.3　运用顺序功能图思想的编程方法

编程之前，先来了解一下顺序功能图的转换规则。这在编程的处理方面至关重要。转换规则如下：

在顺序功能图中，从前一步转换到当前步的转换条件有：

1）转换的前一步必须是活动步。

2）相应的转换条件必须满足。

满足上述两个条件时，前一步可转换到当前步。当前步由“将来时”变为“现在时”，成为活动步。

从当前步转换到后一步时，应满足的条件是：

1）满足转换条件。

2）当前步是活动步。

满足上述两个条件后，当前步成为不活动步，成为“过去时”，程序顺利地进行下一步的操作。

这就像接力比赛一样，如果假想你自己为当前步，当你前面的选手拿着接力棒，那他就是活动步，而当他将接力棒传给你后，你就成为活动步，你前面的人变为不活动步。当你再次将接力棒传递给下一个选手时，你就成为不活动步，而下一个选手成为活动步。你的角色随着接力棒的有无而不断变化，在变化的过程中，程序一步一步地向前运行。

在各种序列的转换方面，单序列中一个转换仅有一个前步和一个后步；而在选择序列的分支和合并处，一个转换也只有一个前步和一个后步，但一个步可能会有多个前步或者多个后步；在并列序列的分支处，转换有几个后步，在转换实现时，应同时将其对应的步（一般为存储位）置位（使这些当前步变为活动步）。在并行序列的合并处，转换有几个前步，只有当它们都变为活动步，并满足转换条件后，才能实现下一步的转换。在转换实现后，应将对应的步（存储位）全部复位（使当前步变为不活动步）。

下面进行编程方法介绍。

当顺序功能图绘制好之后，需要将其翻译成梯形图或者功能块图。即利用顺序功能图思想进行编程。程序包括：控制程序设计和输出程序设计两部分。

控制程序是描述步的行程，用转换条件来控制步的存储位。这些必须符合顺序功能图的转换规则。当某一步为活动步时，对应的存储位置 1，当转换条件满足时，则将其后一步对应的存储位置 1，而当前步则复位为 0。控制电路的梯形图程序如图 6-19 所示。图中显示了前一步（用存储位 M0.2 表示）为活动步时，在转换条件（条件 1 到 n）满足的情况下（即将 M0.2 与条件 1 到 n 相与为真时），当前步（用存储位 M0.3 表示）变为活动步，此时置位当前步（M0.3），复位前一步（M0.2）。当另一转换条件满足时，当前步转换到后一步，此时，置位后一步，复位当前步。

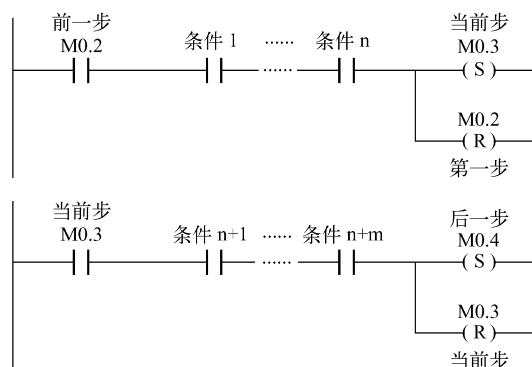



图 6-19　控制电路的梯形图程序

输出程序是描述当前步所执行的动作。由代表步的存储位作为控制接点，输出量是执行器的动作线圈，输出程序相对而言容易设计。如图 6-20 所示为输出程序的梯形图程序。当某一动作在几个步中都执行，则在开始的步中置位此动作，在结束的步中复位此动作。若某一动作只有一步执行，则用输出线圈指令即可。

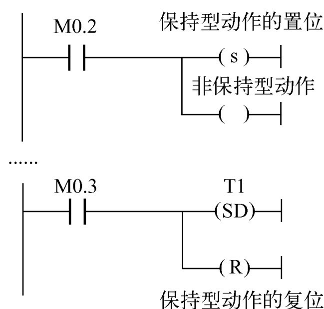



图 6-20　输出程序的梯形图程序

### 1. 单序列的编程方法

这种编程方法中，每一个当前步只有一个前步和一个后步。转换均对应相应存储位的置位和复位。有多少步就有多少个相对应的置位和复位语句。

**例 3**　前述的果汁加工过程的顺序功能图如图 6-21 所示。试进行控制程序和输出程序的梯形图设计。

在 STEP 7 中建立果汁加工过程的符号表，如图 6-22 所示。

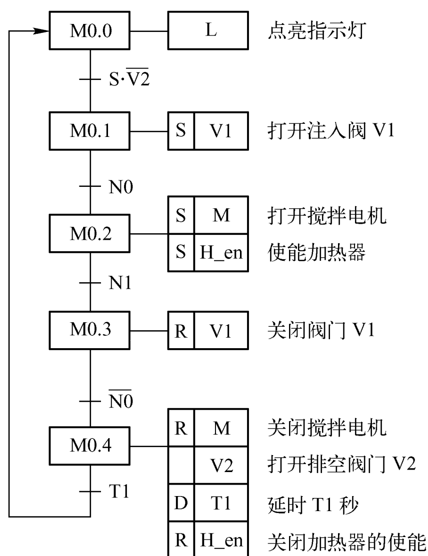



图 6-21　果汁加工过程的顺序功能图

S7 Program(1) (Symbols) -- juice

| | Status | Symbol | Address | Data type | Comment |
|---|---|---|---|---|---|
| 1 | | H | Q 0.3 | BOOL | 加热器 |
| 2 | | H_en | Q 0.5 | BOOL | 加热器使能 |
| 3 | | L | Q 0.0 | BOOL | 指示灯 |
| 4 | | M | Q 0.2 | BOOL | 电动机 |
| 5 | | N0 | I 0.3 | BOOL | 液位高度N0 |
| 6 | | N1 | I 0.2 | BOOL | 液位高度N1 |
| 7 | | S | I 0.0 | BOOL | 启动开关 |
| 8 | | ST | I 0.4 | BOOL | 温度传感器ST |
| 9 | | V1 | Q 0.1 | BOOL | 阀V1 |
| 10 | | V2 | Q 0.4 | BOOL | 阀V2 |

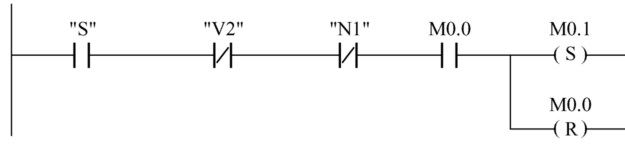
图 6-22　果汁加工过程的符号表

分析：根据单序列的转换条件可知：在前一步变为活动步，且对应的转换条件满足时，转换进行。将当前步设置为活动步，并将前一步复位为不活动步。在设计中，可以将控制程序和输出程序放在一起，也可以放在不同的 FC 功能中。为了能进入循环回路，在 OB100 中设置 M0.0 = 1，即当前步在起始位置 M0.0 处。

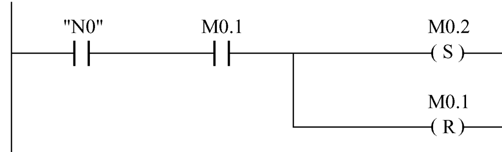
果汁加工系统控制程序的梯形图程序如图 6-23 所示。果汁加工系统输出程序的梯形图程序如图 6-24 所示。

FC 1：果汁加工系统程序

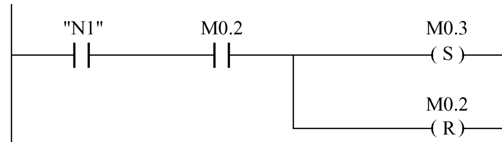
Network 1：灌入果汁



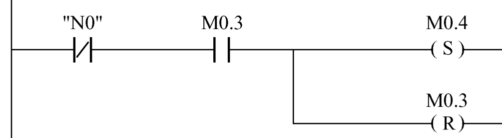

Network 2：搅拌并加热浓缩




Network 3：果汁罐加满




Network 4：排空果汁罐




Network 5：回到起始位置

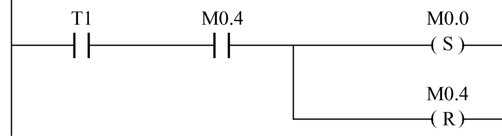
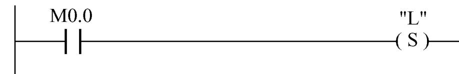
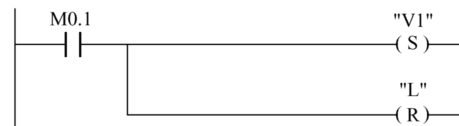



图 6-23　果汁加工系统控制程序的梯形图程序

Network 6：指示灯亮




Network 7：打开阀 V1，加入果汁




Network 8：发动电机，加热器使能

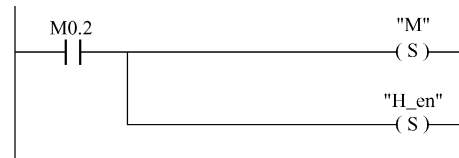



图 6-24　果汁加工系统输出程序的梯形图程序

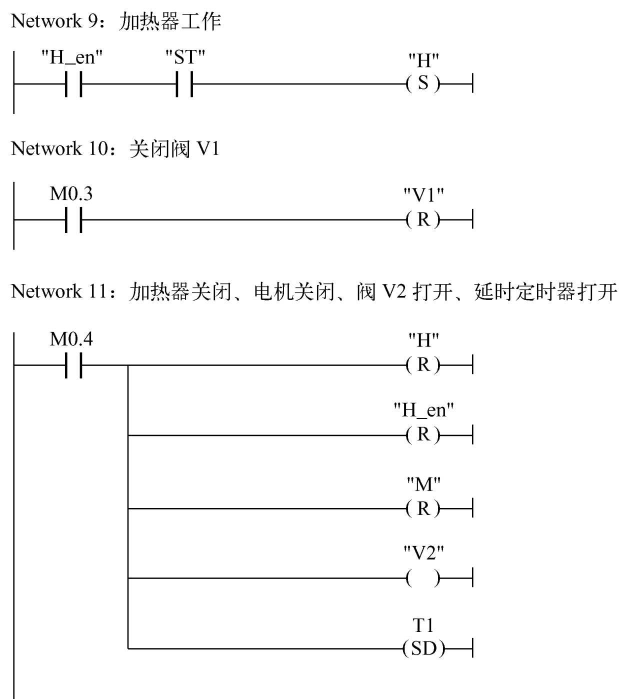



图 6-24　果汁加工系统输出程序的梯形图程序（续）

说明：控制程序从 Network1 始，到 Network5 止，共 5 步，各个步分别用存储位 M0. 0 ~ M0. 4 表示。

在系统满足条件果汁灌入指示灯亮，且起动开关 S 按下时，程序从 M0. 0（起始位置）转换到 M0. 1（灌入果汁）。表现在 Network1 中，将 M0. 0 复位，M0. 1 置位；当果汁高度大于 N0 时，程序从 M0. 1 转换到 M0. 2（搅拌并加热浓缩），表现在 Network2 中，将 M0. 1 复位，M0. 2 置位；当果汁灌满，即果汁高度超过 N1 时，程序从 M0. 2 转换到 M0. 3（果汁罐加满）。表现在 Network3 中，将 M0. 2 复位，M0. 3 置位；当果汁经过加热后，高度浓缩到低于 N0 时，程序从 M0. 3 转换到 M0. 4（排空果汁罐）。表现在 Network4 中，将 M0. 3 复位，M0. 4 置位；当延时时间 30s 到后，程序从 M0. 4 重新转换到 M0. 0（起始位置）。表现在 Network5 中，将 M0. 4 复位，M0. 1 置位。可见，控制程序只考虑步的流程，而对步的动作不做考虑。

输出程序程序从 Network6 始，到 Network11 止。在 Network6 中，在 M0. 0（起始位置）步将允许果汁灌入指示灯 L 点亮；Network7 中，在 M0. 1（灌入果汁）步打开灌入阀 V1，Network8 中，在 M0. 2（搅拌并加热浓缩）步将搅拌电动机 M 起动，并置位使能加热器 H_en；Network9 中，在加热器使能的状态下，当温度低于 80℃时，起动加热器 H。注意：这一环节并没有与某一活动步相连，在满足加热条件时就起动加热器。由于 PLC 的循环扫描，所以能保证加热器在低温时起动，高温时停止。在 Network10 中，在 M0. 3 步（果汁罐加满）关闭灌入阀 V1；在 Network11 中，在 M0. 4 步（排空果汁罐）关闭电动机 M，复位加热器使能 H_en，打开排空阀 V2，打开接通延时定时器 T1。

程序清晰明了，完全不像解析法那样复杂。只要分析清楚系统的顺序控制过程，绘制正

确的顺序功能图，就可以顺利翻译成梯形图程序，基本保证程序的正常运行。

**2. 选择序列的编程方法**

在选择序列中，有分支和合并，某一步有多个后步或多个前步。如图 6-25 所示为选择系列的顺序功能图。

观察此图可以看出，在 M0.0 步之后，有选择序列，它包括两条分支和一个合并。其中转换 I0.0、I0.1 和 I0.2 都只有一个前步和一个后步，比如 I0.2 只有一个前步 M0.0 和一个后步 M0.2，I0.1 和 I0.0 也如此，因此复位和置位存储器位也只需一个。所以，选择序列的分支与合并的编程方法实际上与单序列的编程方法完全相同。

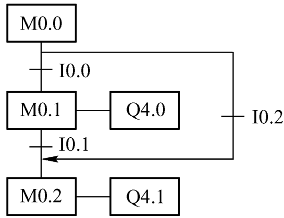



图 6-25 选择序列的顺序功能图

其控制程序如图 6-26 所示。

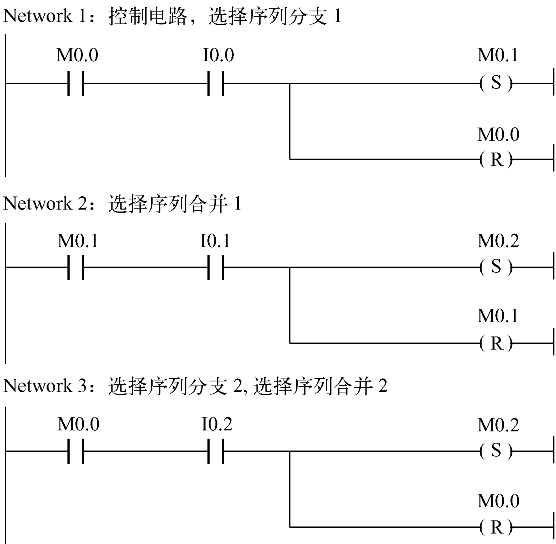



图 6-26 选择序列的控制程序

**例 4** 前述的彩灯系统的顺序功能图如图 6-16 所示。试用选择序列编程方法设计程序。

彩灯显示系统控制程序的梯形图程序如图 6-27 所示。彩灯显示系统输出程序的梯形图程序如图 6-28 所示。

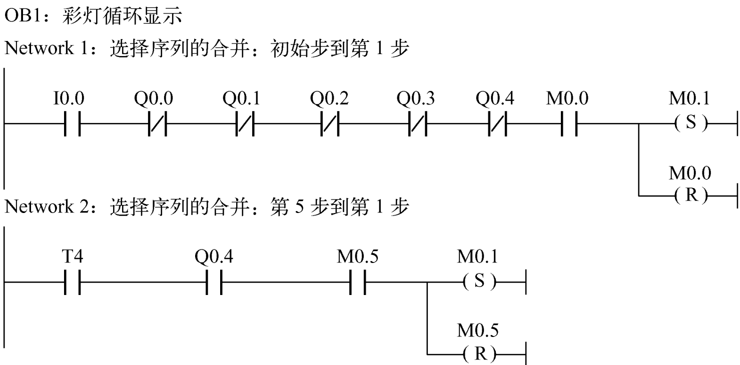
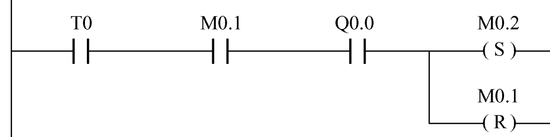
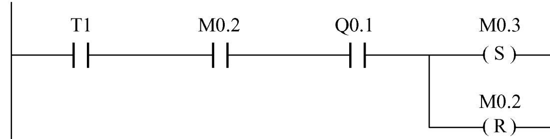



图 6-27 彩灯显示系统控制程序的梯形图程序

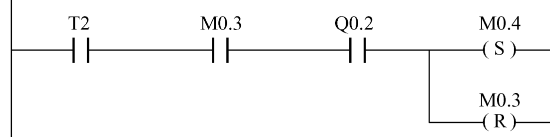
Network 3：第 2 步




Network 4：第 3 步




Network 5：第 4 步




Network 6：第 5 步

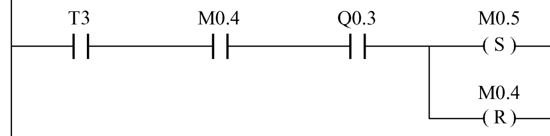
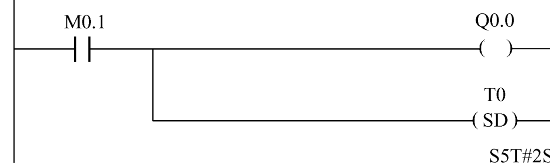
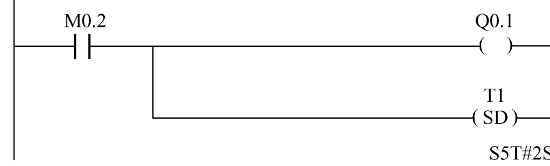



图 6-27　彩灯显示系统控制程序的梯形图程序（续）

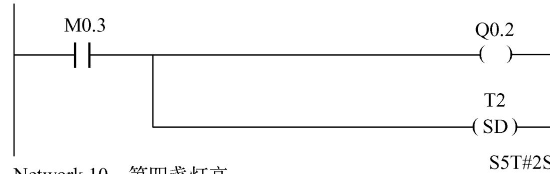
Network 7：第一盏灯亮




Network 8：第二盏灯亮




Network 9：第三盏灯亮




Network 10：第四盏灯亮

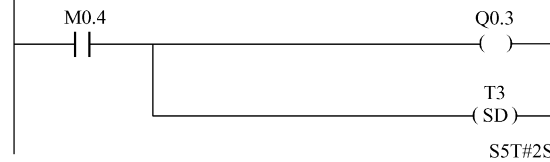



图 6-28　彩灯显示系统输出程序的梯形图程序

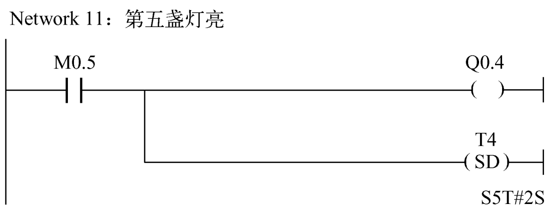



图 6-28　彩灯显示系统输出程序的梯形图程序（续）

说明：控制程序从 Network1 始，到 Network6 止。在 Network1 中，当初始状态五盏灯全都不亮时，此时按下起动按钮，程序从第 0 步转换到第 1 步；在 Network2 中，当第五盏灯亮，并且定时时间 T4 到时，若第 5 步是活动步，则程序从第 5 步转换到第 1 步。由于有两步可以转换到第 1 步，所以属于选择序列的合并；其余各步没有分支，容易理解。

输出程序从 Network7 始，到 Network11 止。在每一步中，点亮一盏灯，同时起动接通延时定时器，定时时间都是 2s。

若要开始循环，必须在 OB100 中设置起始步为 M0.0。

### 3. 并行序列的编程方法

并行序列如图 6-29 所示，图中 M0.2 之后有一个并行序列的分支，在分支的编程中，当 M0.2 成为活动步，并且转换条件 I0.3 满足时，步 M0.3 和 M0.5 同时成为活动步。因此在程序中，要同时将 M0.3 和 M0.5 置位，即同时将分支的活动步置位。在合并的编程中，由于合并必须在 M0.4 和 M0.6 同时成为活动步，并且转换条件 I0.6 满足的情况下实现。所以，在程序中，需将 M0.4、M0.6 以及 I0.6 的常开接点串联，作为后一步置位的条件，当后一步变为当前步时，复位 M0.4 和 M0.6。

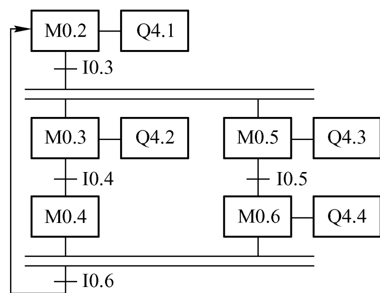



图 6-29　并行序列的顺序功能图

如图 6-30 所示为并行序列的控制程序。

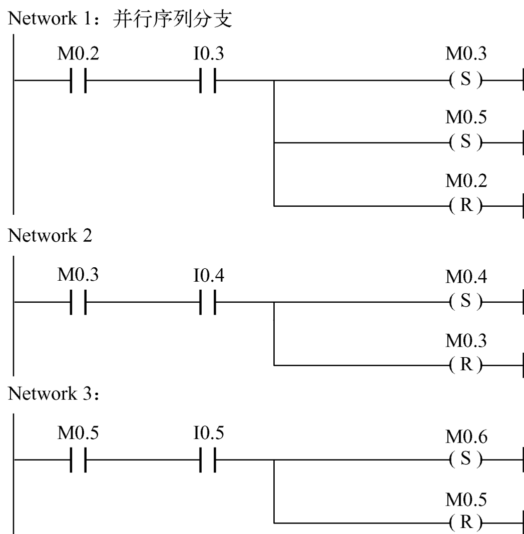



图 6-30　并行序列的控制程序

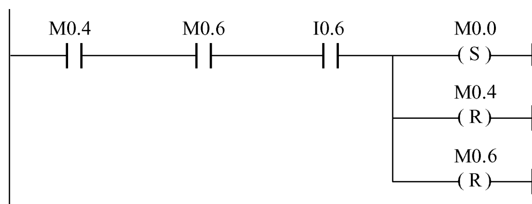



图 6-30　并行序列的控制程序（续）

说明：程序的 Network1 为并行序列的分支段，Network4 为并行序列的合并段。可见有几个并行排列，在分支段程序中就有几个置位语句，同样在合并段程序中对应几个复位语句。

有了上述的分析，我们已经能够处理并行序列的编程。

**例 5**　如图 6-31 所示为专用钻床加工系统示意图，它是用来加工零件的。需加工的零件为圆盘状零件，其上均匀分布了 3 个大孔和 3 个小孔。钻床自动运行的初始状态为：两个钻头在最上位，上限位开关 I0.3 和 I0.5 为 ON。工作过程为：加紧工件，大小钻头开始向下钻孔，至规定的深度后，钻头向上提升并等待，此时工件旋转 120°后，开始加工第二对孔。当 3 对孔加工完后，松开工件，回到初始状态。钻孔的孔数用减计数器来控制，计数器设定初值为 3。试画出顺序功能图，并编写相应的梯形图程序。

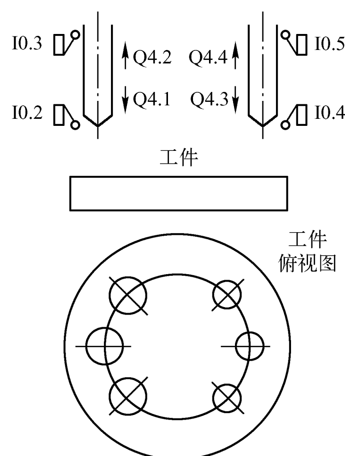



图 6-31　专用钻床加工系统示意图

专用钻床加工系统的变量表如表 6-6 所示。

**表 6-6　专用钻床加工系统的变量表**

| PLC 输入地址 | 变　量　名 | PLC 输出地址 | 变　量　名 |
|---|---|---|---|
| I0. 0 | 起动信号 | Q4. 0 | 夹紧执行 |
| I0. 1 | 工件夹紧 | Q4. 1 | 大钻头钻孔 |
| I0. 2 | 大钻头下限位开关 | Q4. 2 | 大钻头上升 |
| I0. 3 | 大钻头上限位开关 | Q4. 3 | 小钻头钻孔 |
| I0. 4 | 小钻头下限位开关 | Q4. 4 | 小钻头上升 |
| I0. 5 | 小钻头上限位开关 | Q4. 5 | 转盘旋转 |
| I0. 6 | 转盘旋转到位 | Q4. 6 | 松开执行 |
| I0. 7 | 工件松开 | | |

分析：两个钻头向下钻孔和钻头提升的过程用并行序列来表示，在零件没有加工完毕之前，需要重复加工过程；在完成加工后，系统返回初始步。这个过程因为有分支，所以可以用选择序列来表示。专用钻床加工系统的顺序功能图如图 6-32 所示。

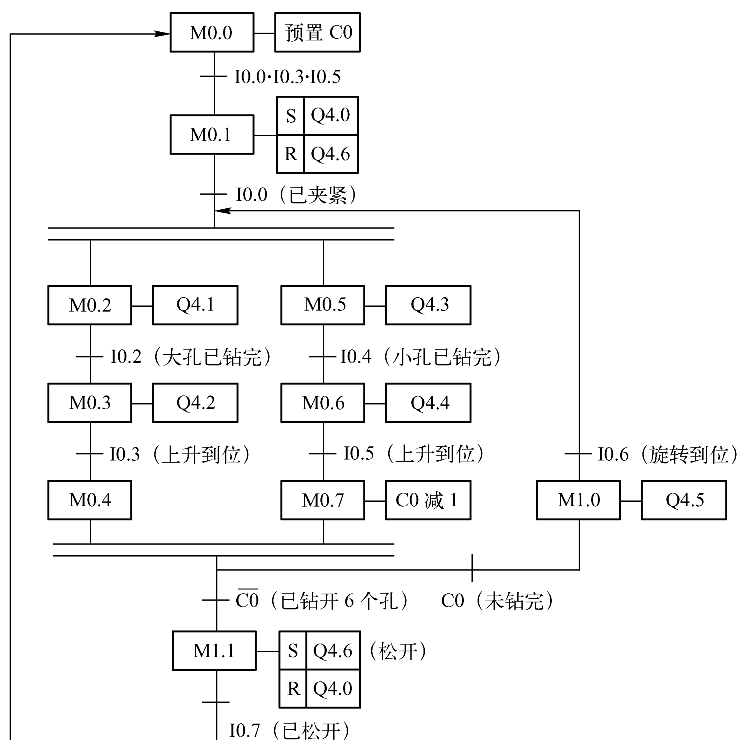



图 6-32　专用钻床加工系统的顺序功能图

钻床加工控制程序的梯形图程序如图 6-33 所示。钻床加工输出程序的梯形图程序如图 6-34 所示。

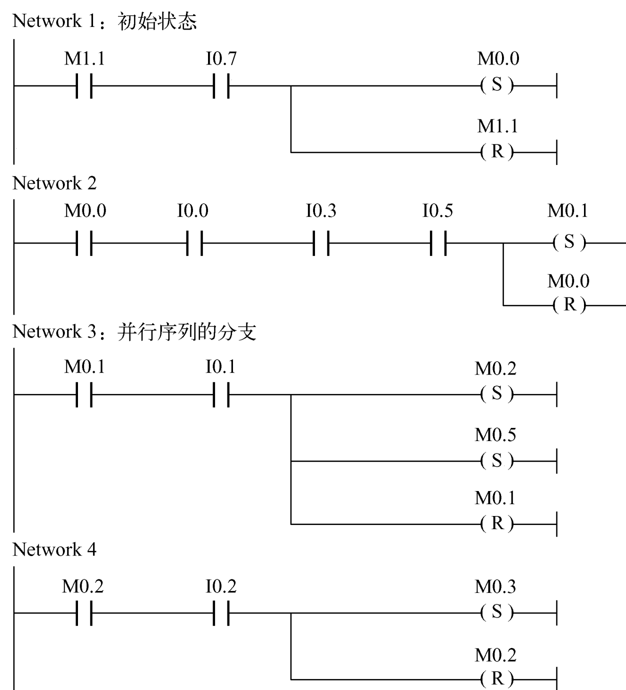



图 6-33　钻床加工控制程序的梯形图程序

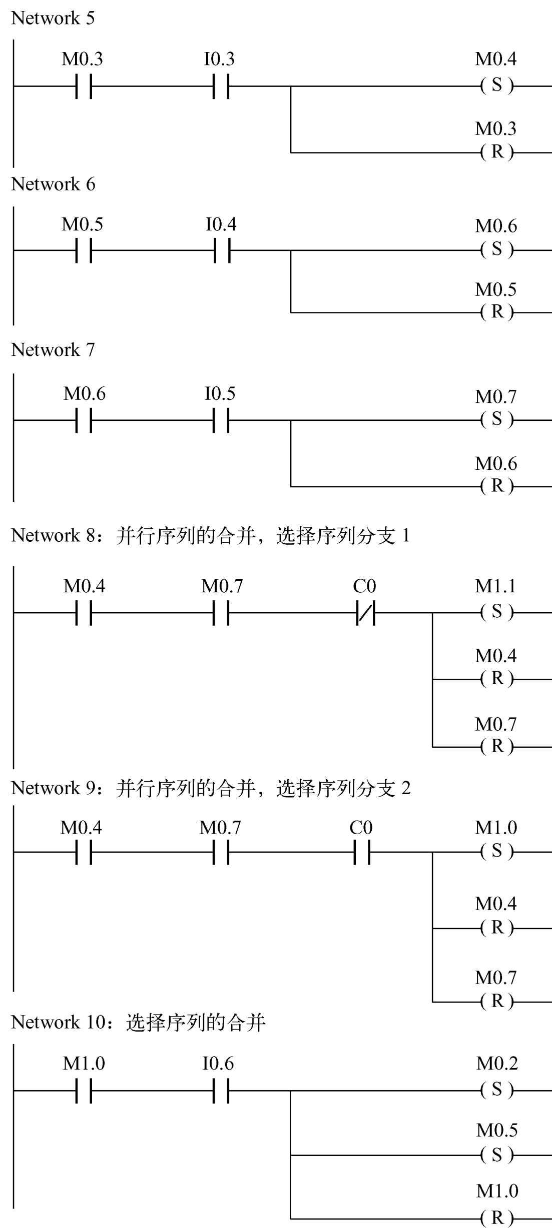



图 6-33　钻床加工控制程序的梯形图程序（续）

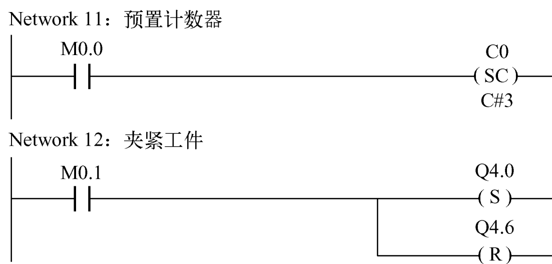



图 6-34　钻床加工输出程序的梯形图程序

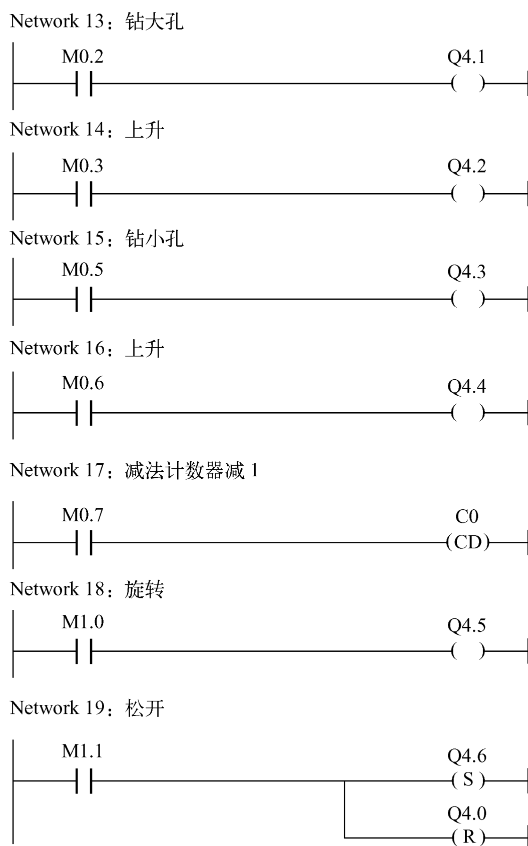



图 6-34　钻床加工输出程序的梯形图程序（续）

在大部分的生产流水线上，顺序控制占据主导地位。如果具有清晰的分析思路，解决问题将游刃有余。以上介绍了各种序列的程序翻译问题，那么如何进行程序调试呢，首先用户可以在仿真 PLC 中将代表起始步的存储位打勾，将相关转换条件（起始步转换到第 1 步的条件）设置好后，观察起始步是否跳转到第 1 步。同理观察其他步的运行情况，是否与顺序功能图的步的转换相同；之后观察相应的动作是否实现。如果步的转换不正确，则在控制程序中查找问题，如果在动作执行中出错，查找输出程序。注意是否存在多线圈问题。同样也可以建立变量表，列出相关步的存储位，列出相关输出动作，将起始步存储位修改成真，结合仿真 PLC，设置好每一步转换的条件，观察代表各步的存储位的变化情况，再观察相应的输出情况。由于有正确的顺序功能图，编程及调试成为易如反掌的事情。

## 6.4.4　具有多种工作方式系统的顺序功能图的编程方法

前面我们曾经提及系统的运行方式有：手动、半自动、自动几种方式。在自动工作方式中又包含连续、单周期、单步、自动返回初始状态几种形式。这些方式均是为了满足生产的需要。而停止方式中，还包括紧急停运（紧急停车）方式，这在系统运行中十分重要。那么这些方式在编程中又是如何实现的呢？下面我们将着重解决这些问题。

**例 6**　如图 6-35 所示是薄形工件存储仓库示意图。通过这个系统的分析，我们可以了解到各种工作方式的解决方法。

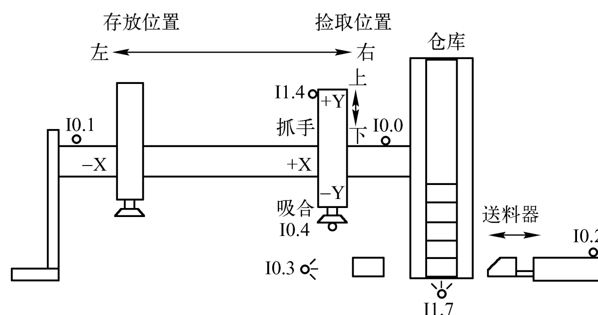



图 6-35　薄形工件存储仓库示意图

**1. 总体描述**

存放薄形工件的仓库是一个圆柱形筒。送料器将一个薄形工件推出至捡取处，在那里抓手将其吸住，并将薄形工件运送至存放处。只要仓库中有薄形工件，则送料器始终工作。

**2. 操作模式**

此系统的操作模式有：

1）回初始模式。

2）手动模式。

3）自动模式。

控制方式的改变通过 WinnCC 所编的操作屏上选择开关来实现转换。

（1）回初始模式

在开始进入单周期、连续和单步工作方式之前，系统必须处于初始位置。如果不满足这一条件，则必须将开关拨到“初始”模式下，使系统自动返回初始位置状态。此例中的初始位置是抓手在上位并在右面。

（2）手动模式

在手动模式下，每个执行器的动作必须按下相应的动作按钮才能实现。如果在硬件的操作面板上没有那么多的按钮，有条件的话可以用软件面板来代替。如图 6-36 所示为 WinCC 操作界面。在其右上方为手动模式方式下的操作指示。如果出现错误操作，则执行器被锁定。如抓手不在上位，则抓手不能从左到右运动。这是为了安全和系统调试的需要。

图 6-36　WinCC 操作界面

在手动编程中，运用经验法就可以实现其功能，但在程序中必须加入联锁功能。如限位开关对运动的极限位置的限制；上行与下行、左行与右行之间的互锁。在转换到其他工作模式时，所有的手动操作必须复位。

（3）自动模式

在自动模式下，程序自动循环，前提条件是各部件位置在其初始位置上。

自动模式下有单周期、单步和连续三种工作方式。这三种方式可用“连续”标志和“转换允许”标志来区分，这些标志以常开接点的方式串联在程序中。其中“连续”标志区分单周期方式和连续工作方式；“转换允许”标志区分单步运行方式和连续自动运行方式。这三种运行方式的编程按照同一个顺序功能图来实现。

“转换允许”标志可以用存储位来表示。它主要是为了区分单步运行和连续运行两种工作模式。“转换允许”常开接点接在控制程序程序的每一步中，相当于每一步多加了一个附加的转换条件。在一般情况下，状态为0，条件不满足，不允许步与步之间的转换。如果状态为1，则在满足转换条件的前提下置位后一步，复位前一步。系统向前步进一步。如图6-35所示为转换允许信号波形图，在连续和单步运行状态下，我们究竟需要什么样的“转换允许”标志信号呢？显然在连续运行模式下，“转换允许”标志必须在起动按钮按下之后，始终为高电平，如图6-37a所示，直至停止按钮按下为止。而在单步运行模式下，“转换允许”标志必须为脉冲信号，如图6-37b所示。在起动按钮（点动按钮）按下的一个PLC扫描周期，“转换允许”为高电平，等起动按钮再次按下，则又有一个周期的高电平产生。这样通过不断按动起动按钮，各个步之间就可以单步运行了。

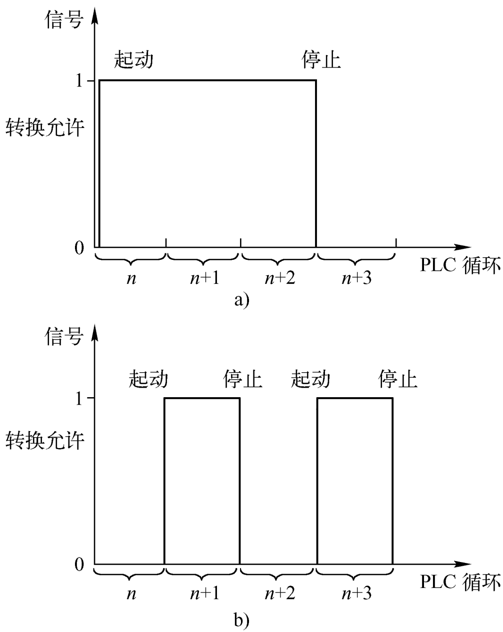


图6-37　转换允许信号波形图

在单周期运行状态下，按一次起动按钮，系统只工作一个周期，所以应该在程序中加入“连续标志”存储器位。当“连续标志”为1时，系统会不断从最后一步返回到第1步，并连续反复地工作。而当“连续标志”为0时，系统返回并停留在初始步（第0步）。完成一个周期的工作。在顺序功能图中表现为选择序列的形式。

**3. 系统分析及设计**

现在我们将实现系统的功能。首先设定系统的初始状态为：抓手在右边、同时在上边，并且没有吸合薄形工件；仓库区中有薄形工件，并且送料器在缩回状态，同时在捡取位置没有薄形工件。

如表6-7所示为薄形工件存储仓库工作符号表。在进行设计之前需要建立符号表或者变量表。

系统的工作过程如下：

表 6-7　薄形工件存储仓库工作符号表

| 设　　备 | 功　　能 | 输　　入 |
|---|---|---|
| B1. 0 | 抓手在右边捡取位置 | I0. 0 |
| B1. 1 | 抓手在左边存储位置 | I0. 1 |
| B3. 1 | 送料器在缩回位置 | I0. 2 |
| B3. 2 | 薄形工件在被捡取的位置 | I0. 3 |
| B4. 1 | 薄形工件在吸合位置 | I0. 4 |
| B5. 1 | 抓手在上位 | I1. 4 |
| B4. 2 | 仓库不空 | I1. 7 |
| S1-4 | 紧急停运按钮 | I1. 3 |
| S1-1 | 手动按钮 | I0. 5 |
| S1-2 | 回初始按钮 | I0. 6 |
| S1-3 | 自动按钮 | I0. 7 |
| S11 | 起动按钮 | I1. 1 |
| S10 | 停止按钮 | I1. 2 |
| S12 | 单步运行按钮 | I1. 5 |
| S13 | 单周期运行按钮 | I1. 6 |
| S20 | 向上按钮 | I2. 0 |
| S21 | 向下按钮 | I2. 1 |
| S22 | 向左按钮 | I2. 2 |
| S23 | 向右按钮 | I2. 3 |
| S24 | 吸合按钮 | I2. 4 |
| S25 | 释放按钮 | I2. 5 |
| S26 | 送料器前推按钮 | I2. 6 |
| 设　　备 | 功　　能 | 输　　出 |
| Y1. 1a | 抓手向右捡取位置运动 | Q4. 0 |
| Y1. 1b | 抓手向左存储位置运动 | Q4. 1 |
| Y3 | 送料器向前运动 | Q4. 2 |
| Y4 | 吸气 | Q4. 3 |
| Y5 | 抓手向下运动 | Q4. 4 |

1）起动按钮按下时，送料器推出薄形工件。
2）送料器撤回。
3）抓手向下运动，吸气（延时）。
4）抓手向上运动。
5）抓手向左运动。
6）抓手向下运动并放下薄形工件，放气（延时）。
7）抓手向上运动。
8）抓手向右运动。
9）回到初始状态。

由于程序结构较复杂，所以需要调用功能来实现。其中功能 FC1 是公用程序，无须调用

条件。在手动工作方式下，调用功能 FC2，调用条件是手动按钮 I0.5。在回初始工作方式下，调用功能 FC3，调用条件是 I0.6，同理在功能 FC4 中实现自动方式和单步运行方式。实现条件是单步运行按钮（I1.5）、单周期运行按钮（I1.6）和自动按钮（I0.7）的并联。在每一段中都加入紧急停运按钮，一旦发生情况，系统无条件停机。主程序 OB1 如图 6-38 所示。

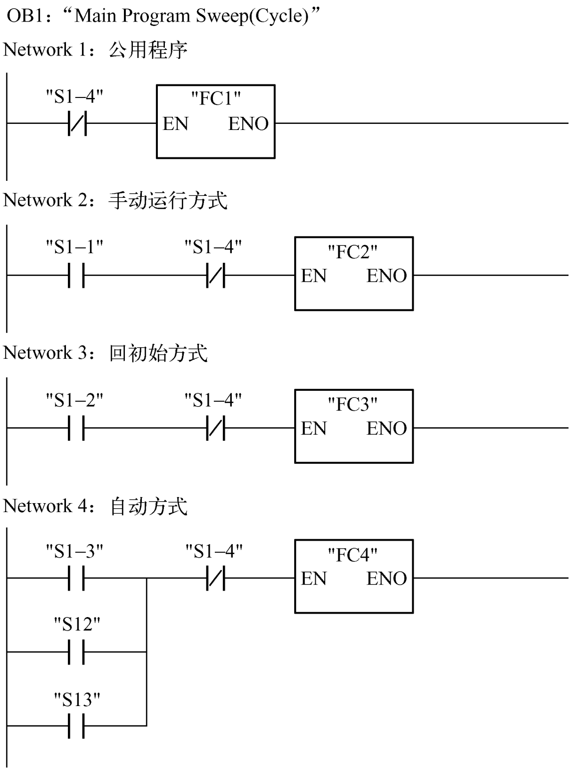



图 6-38　主程序 OB1

首先编写系统的初始化程序 OB100，如图 6-39 所示。在初始化程序中将初始位置加入到其中。设定初始步 M0.0。其中“B1.0”为抓手在右面，“B5.1”为抓手在上位，“B3.2”为捡取位置没有薄形工件，“B4.2”为仓库中不空，“B3.1”为送料器在缩回位置。

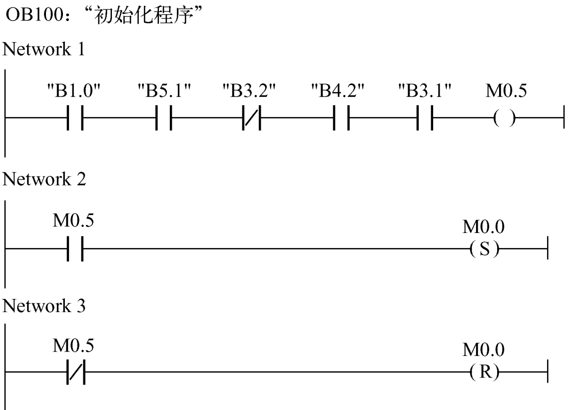



图 6-39　初始化程序 OB100

在公用程序 FC1 中，主要实现一些功能，如系统工作在手动方式和回初始状态下，其处理和在 OB100 中的处理相同，设定了初始状态及 M0.0。这是为什么呢？因为用户在手动模式下运行系统，其中的存储位可能会改变 M0.0 的状态，此时如果没有公用程序中的 M0.0 的设定，再返回自动运行模式，会出现系统无法运行的情况。另一方面，当系统从这两种状态转为自动运行后，为防止表示自动运行方式的存储位出现两个以上的活动步，所以需在此将存储位清零。公用程序 FC1 如图 6-40 所示。可见，公用程序起运行方式转变的过渡作用和保险作用。

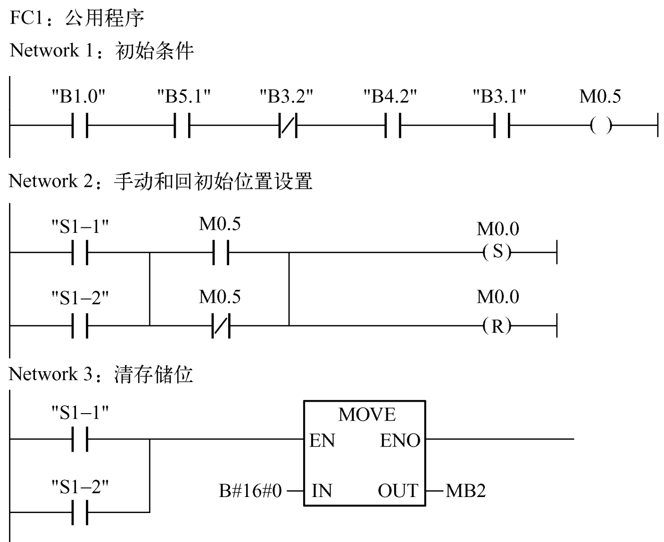



图 6-40　公用程序 FC1

手动程序 FC2 如图 6-41 所示。在 FC2 中，手动操作对应有抓手的上升（S20）、下降（S21）、左行（S22）、右行（S23）、吸合（S24）、释放（S25）、送料器的推出（S26）和退回共 8 个按钮。在手动程序中必须有互锁和限位。比如，向左（Y1.1a）和向右（Y1.1b）运动必须互锁。同时，向左和向右都有限位开关 B1.1 和 B1.0 进行限位。

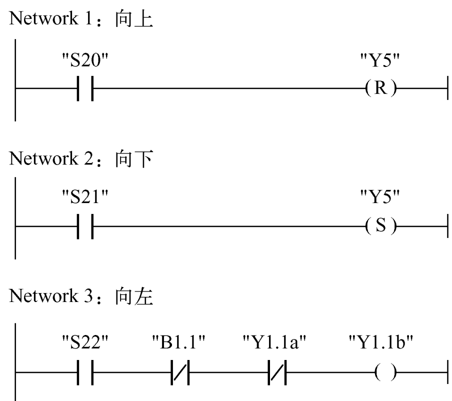



图 6-41　手动程序 FC2

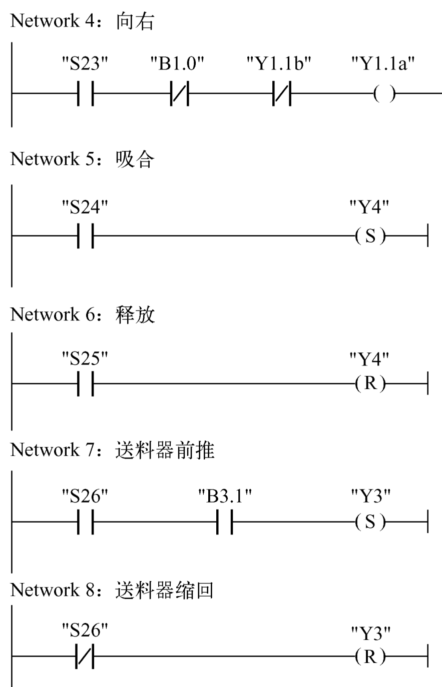



图 6-41　手动程序 FC2（续）

在自动工作状态（包括自动、单步和单周期运行方式）下，考虑到连续运行状态和单步运行状态的不同，在两者的转换中加入“转换允许”存储器位 M0.4。在自动运行方式（S1 -3）下，M0.4 在起动按钮（S11）按下后，始终为高电平。而在单步运行方式下（S12），M0.4 在起动按钮按下时，出现一个扫描周期的高电平，使程序从当前步向后一步运行。

在自动和单周期的转换中加入“连续标志”存储器位 M0.7。在连续标志 M0.7 为 1 时，程序连续自动运行，当 M0.7 为 0 时，程序只工作一个周期，在返回初始状态后停止工作。

当停止按钮（S10）按下时，将存储位清零。如图 6-42 所示为薄形工件系统顺序功能图。

如图 6-43 所示为薄形工件系统的控制程序，图 6-44 所示为薄形工件系统的输出程序。

回初始位置的程序 FC3 大同小异，包括：抓手若在左，则向右运动；抓手若在下，则向上运动；抓手若吸合工件，则放开工件；送料器若在推出位置，则撤回。程序比较简单，可以自己尝试编写。

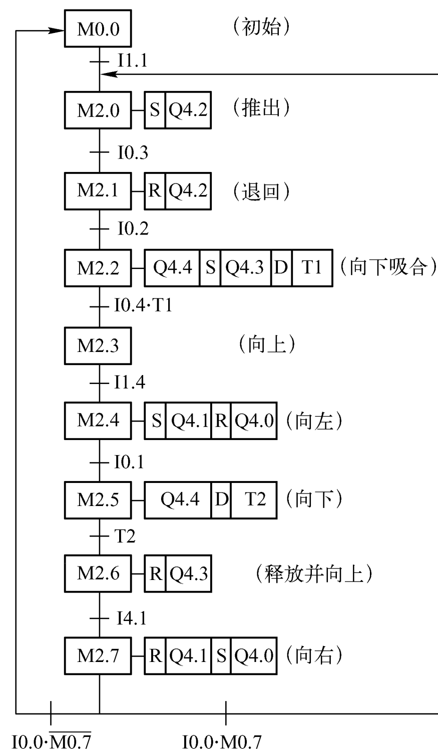
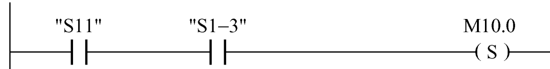


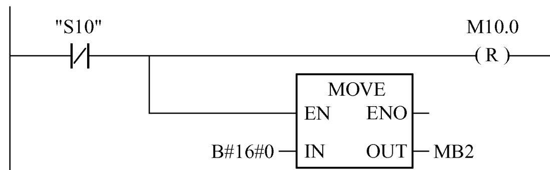

图 6-42　薄形工件系统顺序功能图

FC4：自动、手动和单周期支行方式

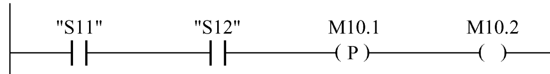
Network 1：自动方式，"S11" 是起动按钮，"S1–3" 是自动运行按钮



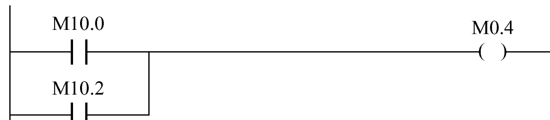

Network 2：停止并复位，"S10" 是停止按钮



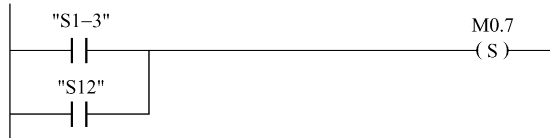

Network 3：单步运行，"S12" 是单步按钮




Network 4：转换标志



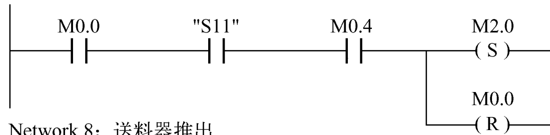

Network 5：连续标志，在自动和单步运行时为 1



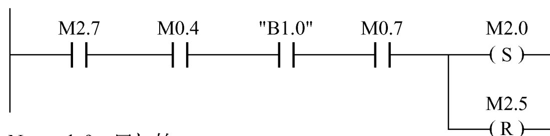

Network 6：连续标志，在单周期运行时为 0



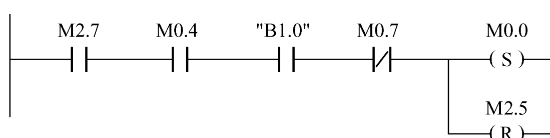

Network 7：进入循环体




Network 8：送料器推出




Network 9：回初始




Network 10：送料器缩回

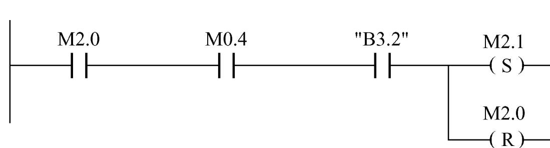
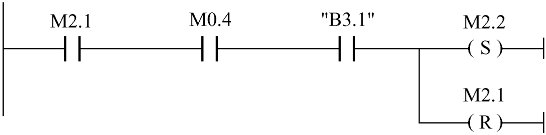
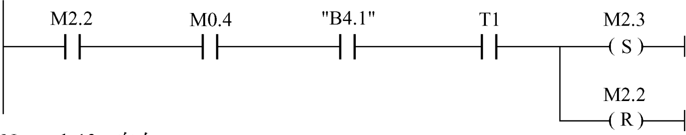



图 6–43　单步、单周期、自动方式下薄形工件系统的控制程序

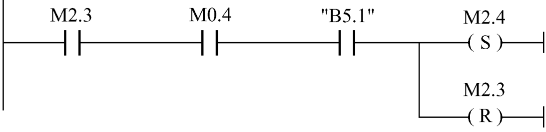
Network 11：抓手向下，吸合工作



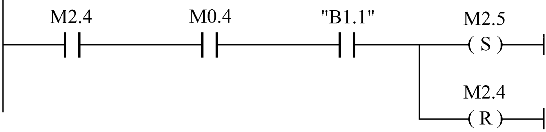

Network 12：抓手向上



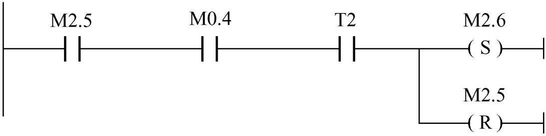

Network 13：向左




Network 14：向下




Network 15：释放并向上




Network 16：向右

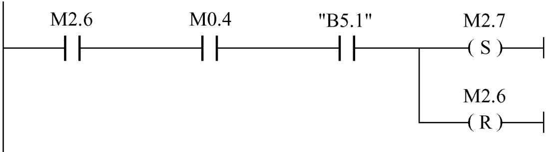
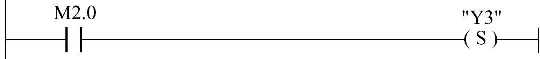
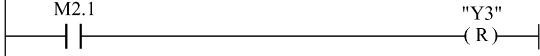



图 6-43 单步、单周期、自动方式下薄形工件系统的控制程序（续）

Network 17：输出电路：推出




Network 18：缩回




Network 19：向下吸合




图 6-44 单步、单周期、自动方式下薄形工件系统的输出程序

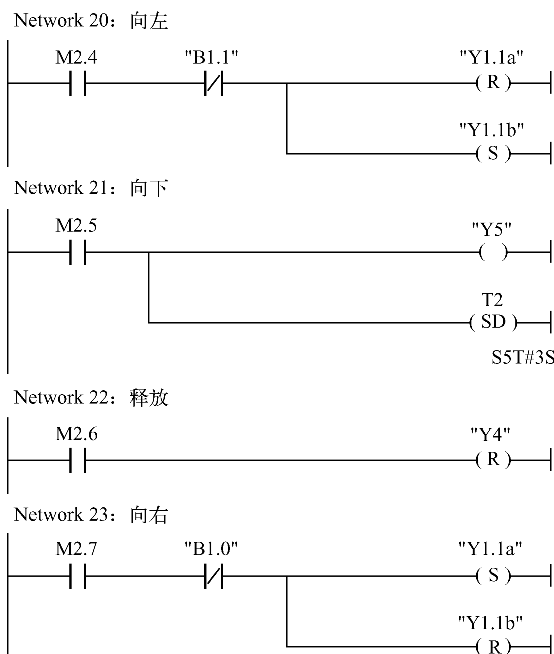



图 6-44　单步、单周期、自动方式下薄形工件系统的输出程序（续）

## 习题 I

1. 运料小车运行过程如图 6-45 所示。小车原位在左边（SQ1），当按下起动按钮 SB，小车前进。当运行至料斗下方（SQ2）时，料斗打开给小车加料，延时 8s 后料斗关闭。小车后退返回至 SQ1 处，打开小车底门卸料，6s 后卸料完毕，如此循环下去。试自行建立变量表，并用顺序功能图设计此系统。

2. 有一个选择性分支的顺序功能图如图 6-46 所示，试用梯形图程序实现之。

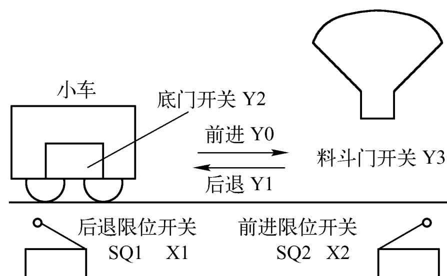



图 6-45　习题 1 图

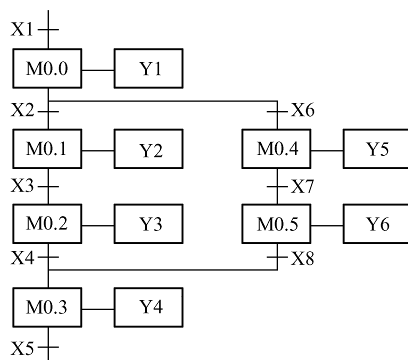



图 6-46　习题 2 图

3. 有一个并行序列的顺序功能图如图 6-47 所示，试用梯形图程序实现之。

4. 设计机械手控制系统。如图 6-48a 所示是机械手的工作示意图。其操作面板如图 6-48b 所示。可以看出，此系统包括多种运行方式，如手动方式、自动方式。而自动方式又包括连续、单周期、单步、自动返回初始状态几种工作方式。其中手动方式及回初始方

式比较简单，用经验法编程即可。而单步、单周期、连续方式均需要绘制顺序功能图。机械手的初始位置是：机械手在最上位和最左边且机械手处于松开状态。设计时，可以将公用程序放在 FC1 中，手动程序放在 FC2 中，连续单周期、单步运行放在 FC3 中，回初始程序放在 FC4 中。注意在编程之前，要求建立符号表。

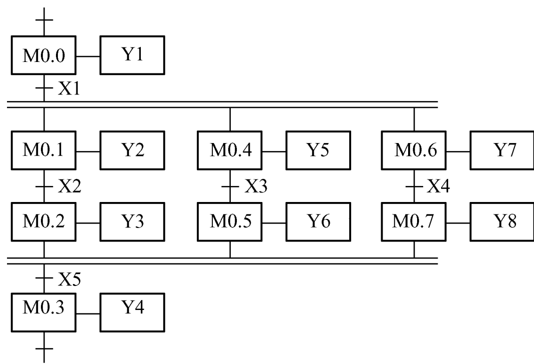



图 6-47　习题 3 图

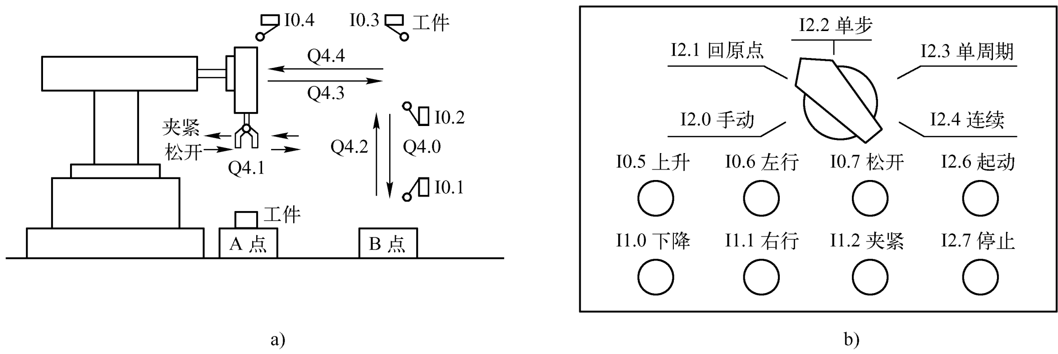



图 6-48　习题 4 图

5. 液压滑台的进给运动示意图如图 6-49 所示。设动力滑台在初始位置时停在左边，限

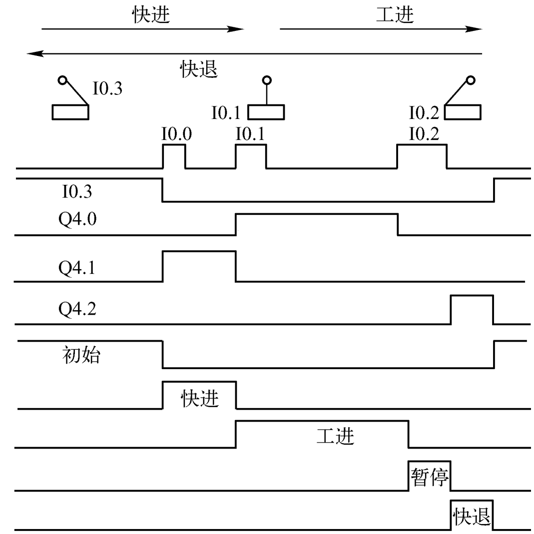



图 6-49　习题 5 图

位开关 I0.3 为 ON，Q4.0～Q4.2 是动力滑台的 3 个电磁阀（分别控制液压滑台的工进、快进、快退）。当按下起动按钮后，动力滑台开始工作，其工作周期由快进、工进、暂停和快退组成。试绘制顺序功能图，并进行梯形图程序。

6. 试编写十字路口交通灯程序，其顺序功能图如图 6-18a、b 所示。

### 6.4.5 MPS 工作站的设计

德国 Festo 教学仪器制造厂生产的 MPS 工作站采用现代气动技术、传感器技术及 PLC 控制技术，对生产线进行了模块化及标准化设计，由标准化 MPS 气动元件组成大型的加工装配生产线。模块包括：供料站、检测站、加工站、提取站和分类站等。模块间通过 PROFIBUS 现场总线（或光电传感器）互相通信。

**1. MPS 供料站描述**

如图 6-50 所示为供料站组成。供料站共包括两个模块：

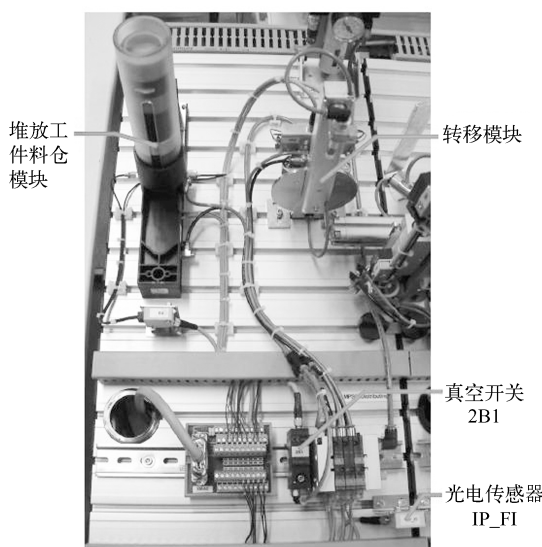



图 6-50　供料站组成

1）堆放工件料仓模块。

2）转移模块。

工件料仓桶中最多可以放入 8 个工件，每个工件由桶的上部开口处放入，桶内是否有工件由透光传感器 B4 来检测。双作用汽缸推出最低部的工件至传输平台。推出汽缸的位置由光电传感器 1B1、1B2 来探测。如图 6-51 所示为供料站传感器位置。如图 6-52 所示为供料站执行机构及传感器安装位置。

转移模块通过真空吸盘吸合被分捡出的工件。真空开关 2B1 检查是否工件被吸住。之后，由摆动气缸的传输臂将工件从堆放料仓 3S1 处转移到下一站的工件托盘 3S2 处。

初始状态：

1）推出汽缸在缩进位置。

2）旋转臂在堆放料仓位置。

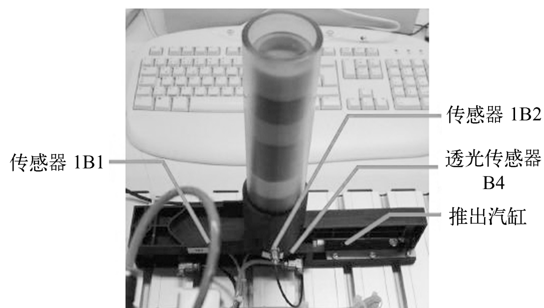



图 6-51　供料站的传感器位置

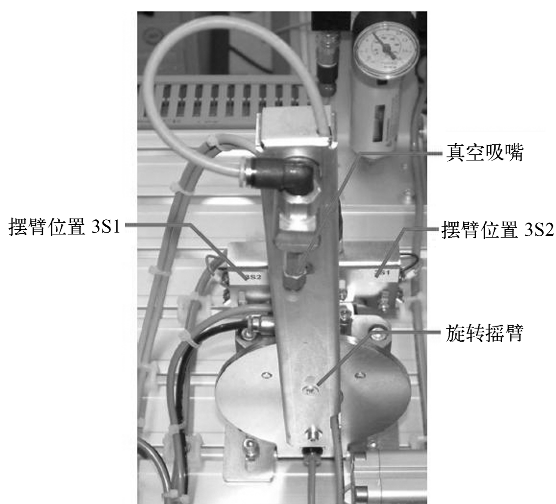



图 6-52　供料站执行机构及传感器安装位置

工作过程：

1）如果料仓中有工件，并且起动按钮按下时，旋转臂旋转到下一站工件托盘处。

2）推出汽缸将工件推出。

3）旋转臂旋转到料仓位置。

4）真空阀打开，工件被牢牢吸合。

5）推出汽缸缩回。

6）旋转臂旋转到下一站工件托盘位置。

7）真空阀关闭，工件被释放落在下一站工件托盘中。

8）旋转臂再次旋转到料仓位置，如此循环。

MPS 工作站有简单的操作面板，包括人工操作模式、自动操作模式和回初始位置。用户还可以用 WinnCC 设计自己的操作面板。

回初始位置模式：在开始任何自动循环之前，操作单元必须在其初始位置。

在单步执行模式，每按动起动按钮一下，操作单元单步执行动作一步。

人工操作模式：在人工操作模式下，每一个执行器必须通过按动相应动作按钮才能被激活，这样做是为了避免错误的操作。

自动操作模式：当操作单元位于初始状态下，按下起动按钮，系统开始自动循环扫描。当停止按钮按下时，自动循环扫描过程结束。

**2. 系统设计**

在系统设计中，我们仅对自动模式下的单步运行方式和连续运行方式作了分析。

首先，编程前先定义符号表，如图 6-53 所示为供料站符号表。

如图 6-54 所示为供料站的顺序功能图。从顺序功能图可以看出：某些动作有时在两步甚至多步中出现，此时要注意输出程序的编写中必须避免多线圈的问题（第 4 章提及），即将相同动作的输出并联处理。或者在顺序功能图的绘制中将动作开始的步用置位语句将其置位，在动作结束的步中用复位语句将其复位。两种绘制方法均可。

01VE_KFA (Symbole) -- mps_control

| | Status | Symbol | Address | | Data type | Comment |
|---|---|---|---|---|---|---|
| 1 | | 1B1 | I | 0.2 | BOOL | 推出气缸在推出位置 |
| 2 | | 1B2 | I | 0.1 | BOOL | 推出气缸在缩回位置 |
| 3 | | 1Y1 | Q | 0.0 | BOOL | 推出气缸推出工件 |
| 4 | | 2B1 | I | 0.3 | BOOL | 工件被吸住 |
| 5 | | 2Y1 | Q | 0.1 | BOOL | 真空泵开启 |
| 6 | | 2Y2 | Q | 0.2 | BOOL | 真空泵关闭 |
| 7 | | 3S1 | I | 0.5 | BOOL | 旋转臂在料仓处 |
| 8 | | 3S2 | I | 0.4 | BOOL | 旋转臂在工件托盘处 |
| 9 | | 3Y1 | Q | 0.3 | BOOL | 旋转臂向料仓处旋转 |
| 10 | | 3Y2 | Q | 0.4 | BOOL | 旋转臂向工件托盘处旋转 |
| 11 | | B4 | I | 0.6 | BOOL | 仓库空传感器 |
| 12 | | IP_FI | I | 0.7 | BOOL | 下一站准备就绪 |
| 13 | | S1 | I | 1.0 | BOOL | 起动按钮 |
| 14 | | S2 | I | 1.1 | BOOL | 停止按钮（常闭） |
| 15 | | S3 | I | 1.2 | BOOL | 自动/单步转换开关 |
| 16 | | S4 | I | 1.3 | BOOL | 复位开关 |

图 6-53　供料站符号表

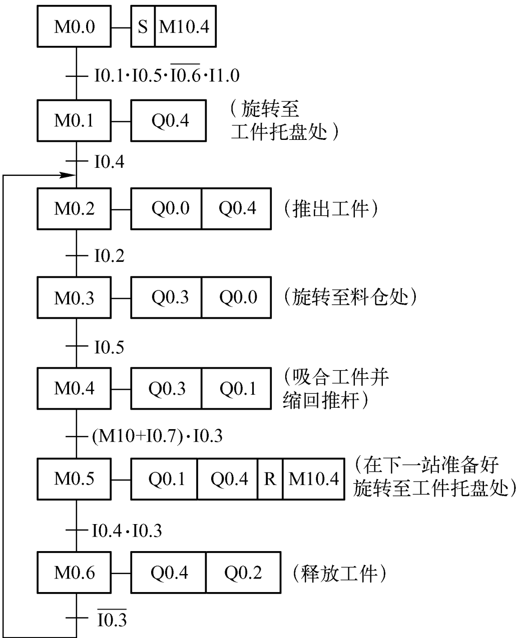



图 6-54　供料站的顺序功能图

编程时，将控制程序放在 FC1 程序中，而把输出程序放在 FC2 中。供料站控制程序如图 6-55 所示。供料站输出程序如图 6-56 所示。

FC1：控制回路程序

Network 1：复位，初始化存储位

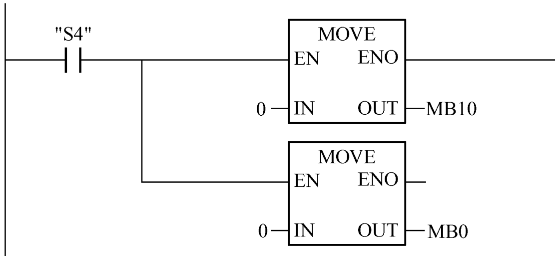
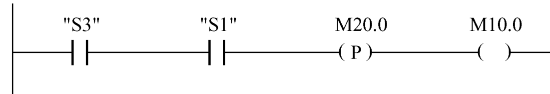
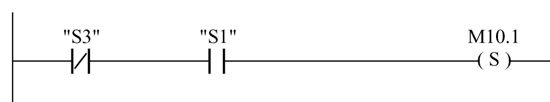



图 6-55　供料站控制程序

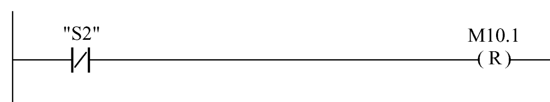
Network 2：单步运行模式



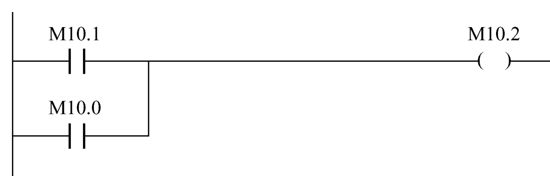

Network 3：自动运行模式



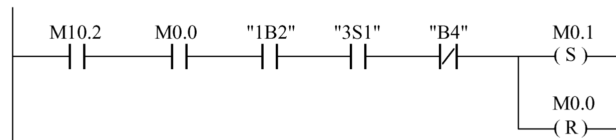

Network 4：停止



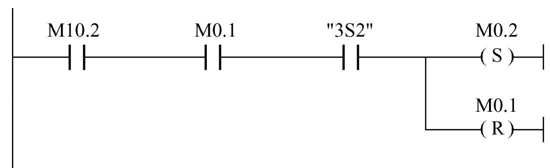

Network 5：设置转换标志



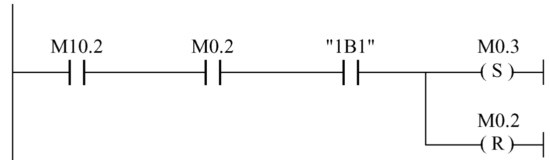

Network 6：若在起始位置，则旋转至工件托盘处




Network 7：推出工件




Network 8：旋转至料仓处




Network 9：吸合工件，推动活塞杆缩回

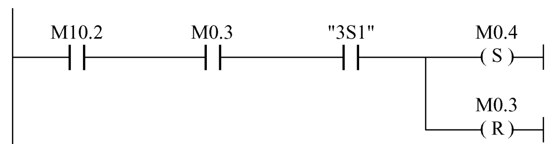
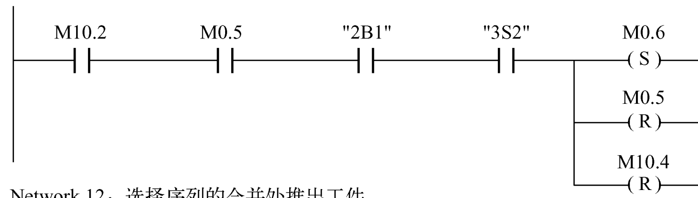



图 6-55　供料站控制程序（续）

Network 11：释放工件




Network 12：选择序列的合并处推出工件

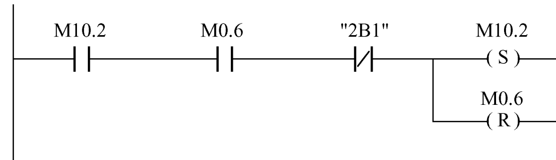
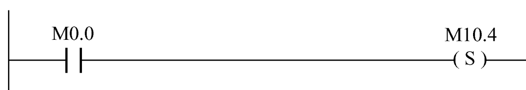


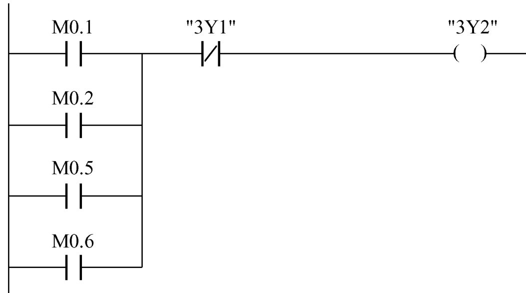

图 6-55　供料站控制程序（续）

FC2：输出电路程序

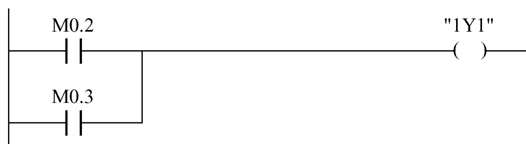
Network 1：第一次运行，设置通过信号



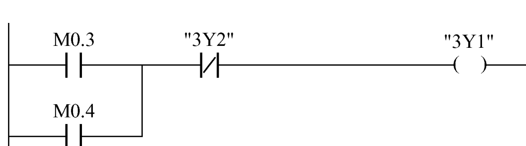

Network 2：旋转臂向工件托盘处旋转




Network 3：推出活塞推出工件




Network 4：旋转臂向料仓处旋转




Network 5：真空泵开启

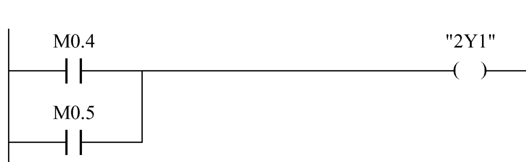



图 6-56　供料站输出程序

Network 6：真空泵关闭，关闭通过信号

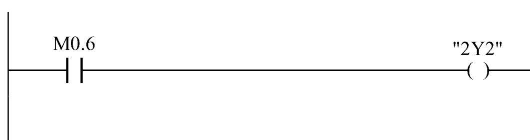



图 6-56　供料站输出程序（续）

说明：控制程序从 Network1 起，到 Network12 止。其中 Network1 为复位按钮的作用，将相关存储器清零；Network2 中，为单步运行设置转换标志；Network3 和 Network4 中，为连续运行设置转换标志；Network5 将两者合一，设置了最终的转换标志 M10.2；并将其串联于其后的控制程序中。Network6 在满足初始状态的条件下，程序从 M0.0 转换到 M0.1，并执行其相对应的动作：旋转臂旋转至工件托盘处（让出位置，使工件可以从料仓中推出）；其后程序，以此类推。注意：在程序的 Network10 中，当下一站准备好后（工件托盘处没有工件），则旋转臂向工件托盘处转动。当工作站刚开始工作时，下一站准备好的信号没有产生，所以预先设置第一次通过信号 M10.4。在下一站正常工作，可以发送通信信号 I0.7 后，将其取消。注意：在下一站准备好后，下一站发出信号 Q0.7，本站 I0.7 接收此信号后，常闭触点接通。

输出程序的第 0 步，设置通过信号，当旋转臂携带工件旋转至工件托盘处，下一站可以工作时，在第 6 步取消通过信号。注意：前面提到的多线圈问题在输出程序的 Network2 至 Network5 中都存在。所以，必须并联处理，否则由于循环扫描的缘故，后面的程序会刷新前面的结果，导致程序出错。

为了程序的正常运行，在 OB100 中设置初始步为 M0.0。

**3. 检测站描述**

如图 6-57 所示为检测站的组成。检测站包括 5 个模块。

1）识别模块。

2）提升模块。

3）测量模块。

4）气垫滑动模块。

5）滑动模块。

检测站决定工件的内部特性。检测站的传感器位置如图 6-58 所示。

识别模块通过两个具有数字输出的接近传感器来识别工件的颜色。这两个接近传感器，一个为电容传感器、另一个为光电式传感器。其中电容式传感器 Part_AV 能检测到金属、红色和黑色工件（检测有无工件）；光电式传感器能检测黑色工件。两个传感器同时工作可以检测出“黑”和“非黑”工件。

在工件被提升之前，反射式传感器监控工作区域的工件托盘是否是空的。

来自识别模块的工件被提升模块提升到测量模块位置，使用的执行器是无推杆汽缸。

无推杆汽缸的位置由感应接近传感器 1B1 和 1B2 来检测，如图 6-59 所示。

推出汽缸的位置由磁接近传感器 2B1 来检测。

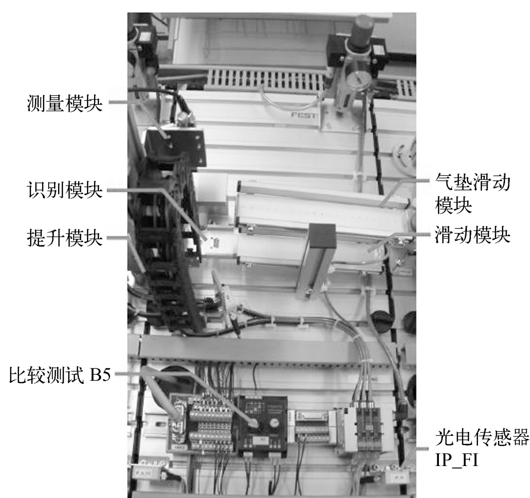



图 6-57　检测站的组成




图 6-58　检测站的传感器位置

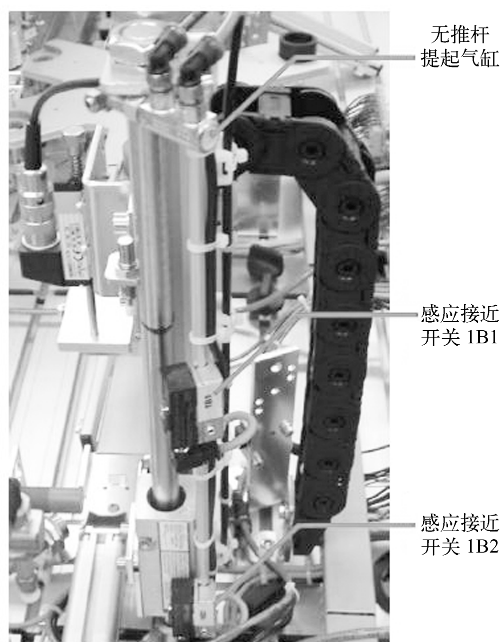



图 6-59　检测站的执行机构及传感器位置

测量模块包括测量工件高度的模拟传感器，模拟测量值可以通过模拟比较值转换成数字量。它能测量工件的高度是 2.5mm 高，此时数字量输出为 1。低于 2.5mm，输出为 0。本系统的红色和金属工件具有 2.5mm 的高度，而黑色工件低于此高度。

测量之后，通过识别模块的活塞推出的红色和金属工件被输送到气垫滑动模块。而气垫滑动模块将工件传输到下一站。同样通过识别模块的活塞推出的错误工件被输送到底部安装的滑动模块。这样合格的工件被挑选出来。

开始需求：

1）工件在工件托盘中。

2）工作区是空的（旋转臂不在工件托盘处）。

初始位置：

1）提升汽缸在底部。

2）推出汽缸在缩回位置。

3）气垫滑动模块在关闭状态。

工作过程：

1）检测工件的颜色和材料。

2）提升汽缸上升。

3）测量工件的高度。

如果测试正确：

1）打开气垫滑动模块。

2）将推出汽缸的活塞杆推出。

3）将推出汽缸的活塞杆缩回。

4）关闭气垫滑动模块。

5）将提升汽缸下降。

6）返回初始状态。

如果测试失败：

1）将提升汽缸下降。

2）将推出汽缸的活塞杆推出。

3）将推出汽缸的活塞杆缩回。

4）返回初始位置。

**4. 工作站设计**

在编程之前，首先编写符号表，检测站的符号表如图 6-60 所示。

02PR_KFA (Symbole) -- mps_control

| | Status | Symbol | Address | Data type | Comment |
|---|---|---|---|---|---|
| 1 | | 1B1 | I 0.4 | BOOL | 提升气缸在高位 |
| 2 | | 1B2 | I 0.5 | BOOL | 提升气缸在低位 |
| 3 | | 1Y1 | Q 0.0 | BOOL | 提升气缸下降 |
| 4 | | 1Y2 | Q 0.1 | BOOL | 提升气缸升起 |
| 5 | | 2B1 | I 0.6 | BOOL | 推出气缸在缩回位置 |
| 6 | | 2Y1 | Q 0.2 | BOOL | 推出气缸的活塞杆推出 |
| 7 | | 3Y1 | Q 0.3 | BOOL | 气垫滑动打开 |
| 8 | | B2 | I 0.1 | BOOL | 测试黑工件 |
| 9 | | B4 | I 0.2 | BOOL | 检测旋转臂是否工件容納器中,空为1 |
| 10 | | B5 | I 0.3 | BOOL | 工件高度正确 |
| 11 | | IP_F | I 0.7 | BOOL | 下一站准备就绪 |
| 12 | | IP_N_FO | Q 0.7 | BOOL | 本站就绪 |
| 13 | | Part_AV | I 0.0 | BOOL | 工件检测 |
| 14 | | S1 | I 1.0 | BOOL | 起动按钮 |
| 15 | | S2 | I 1.1 | BOOL | 停止按钮 |
| 16 | | S3 | I 1.2 | BOOL | 自动/单步按钮 |
| 17 | | S4 | I 1.3 | BOOL | 复位按钮 |

图 6-60　检测站的符号表

按照工作过程绘制出检测站的顺序功能图，如图 6-61 所示。

将检测工作站控制程序放在 FC1 程序中，而把输出程序放在 FC2 中。检测站控制方式设定的梯形图程序如图 6-62 所示。检测站控制程序如图 6-63 所示。检测站输出程序如图 6-64 所示。

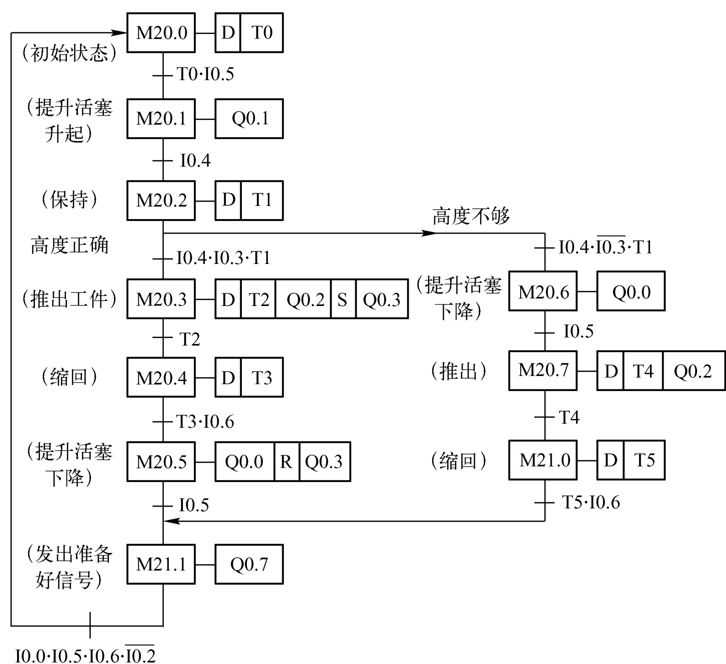
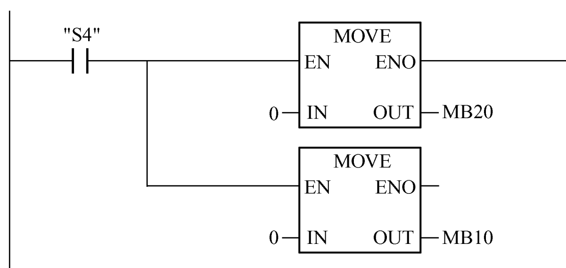


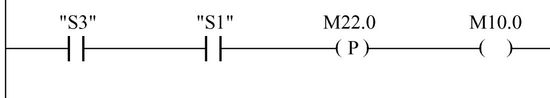

图 6-61　检测站的顺序功能图

FC1：控制电路程序

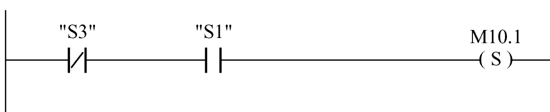
Network 1：复位，初始化存储位




Network 2：单步工作方式




Network 3：自动工作方式




Network 4：停止

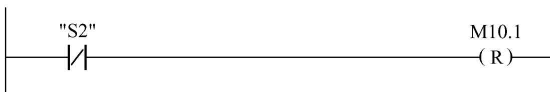


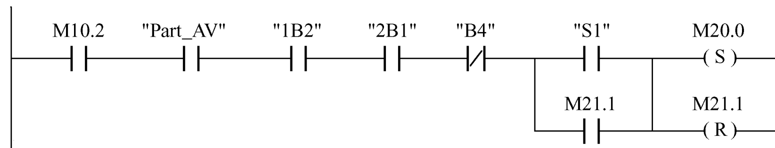

图 6-62　检测站控制方式设定的梯形图程序

Network 5：在单步和自动方式下，设置转换标志 M10.2

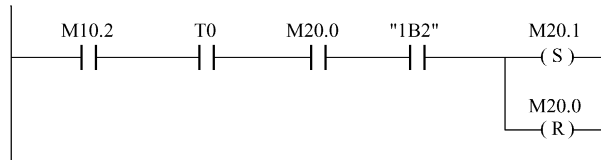



图 6-62　检测站控制方式设定的梯形图程序（续）

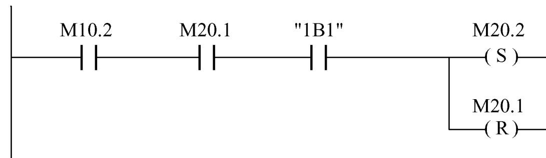
Network 6：初始状态，起动定时器 T0（选择序列合并）



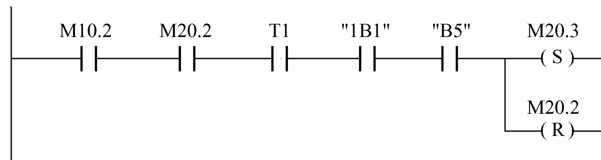

Network 7：等待前一站的旋转臂移走后工件被提起




Network 8：保持并等待测高




Network 9：检测高度正确则推杆推出，气垫滑动打开，
并起动定时器 T2（选择序列分支）




Network 10：定时时间到，推杆缩回，起动定时器 T3




Network 11：定时时间到，关闭气垫滑动，提升活塞下降

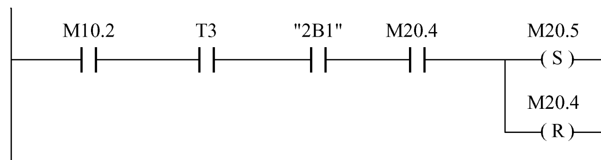
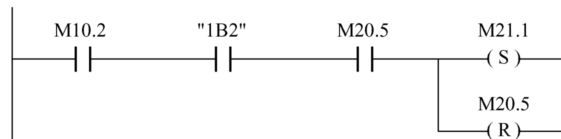
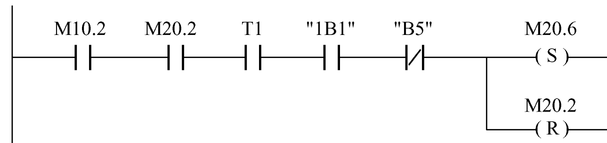



图 6-63　检测站的控制程序

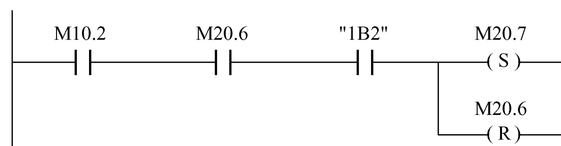
Network 12：下降至低位，保持并发出通信信号



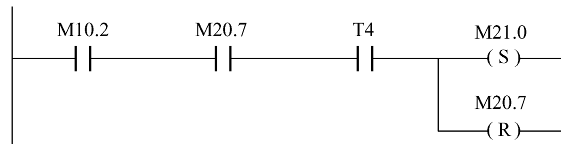

Network 13：高度不正确，下降（选择序列分支）




Network 14：下降至低位，保持并推出，同时起动定时器 T4




Network 15：推杆缩回，同时起动定时器 T5




Network 16：选择序列的合并，发出通信信号

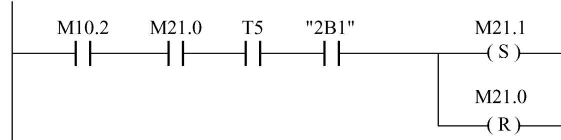
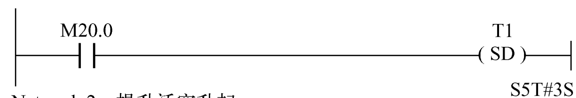


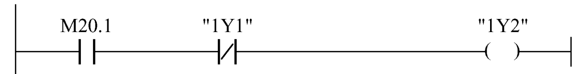

图 6-63　检测站的控制程序（续）

FC2：输出电路程序

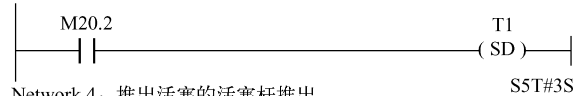
Network 1：起动定时器 T0




Network 2：提升活塞升起




Network 3：起动定时器，等待测高




Network 4：推出活塞的活塞杆推出

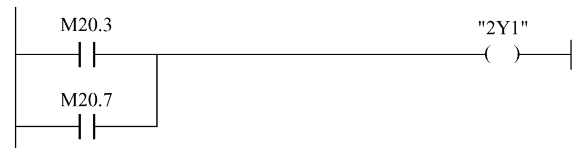



图 6-64　检测站的输出程序

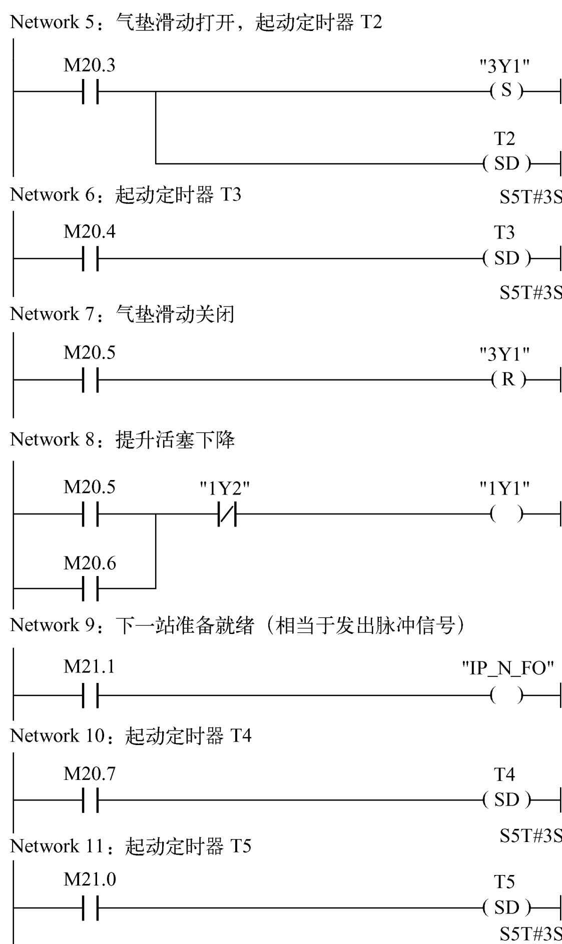



图 6-64　检测站的输出程序（续）

在两站的编程中关键是两个站之间的通信，当站 2 的提升汽缸在低位时，工件托盘在低位，处于可以容纳工件的位置。此时发送脉冲信号 IP_N_FO 给站 1，这样站 1 接收到这个信号（站 2 发输出信号：IP_N_FO，站 1 的输入信号：IP_FI 接收信号）后，等待在料仓处的旋转臂开始吸着工件旋转，到工件托盘处放下工件（如果不这样做，结果是提升汽缸空着上升）。之后站 1 的旋转臂开始往料仓处旋转。旋转需要一定时间，所以在站 2 的起始位置起动定时器 T0，等待旋转臂的旋转，旋转结束后，站 2 检测到工件，返回到初始状态，延时后，提升汽缸才开始带动工件托盘上升。

编程中还需注意几点细节：

1）如站 2 的提升汽缸在低位和高位处需停止，否则，如果一直给提升汽缸加信号，会出现提升汽缸在高位时，冲顶上部的传感器，进而使提升汽缸出现故障，磁耦合脱落。

2）在开始工作时，站 1 开始的动作是旋转臂旋转到工件托盘处，这样，料仓处的汽缸才可能推出工件。而站 2 的初始工作条件是旋转臂不在工件托盘所在的工作区，那么两者如何协调呢？因为站 1 开始工作时，站 2 并不能立即开始工作，因工件托盘里没有工件。所以，两个站的起动时间并不相同，因此实际上并不需要两个站在时间上的进行协调。

MPS 工作站还有其他工作站，如加工、提取和分类站等构成装配生产线。其他站的工作大同小异，这里不再赘述。

## 6.4.6 GRAPH 编程

在 STEP 7 V5.4 以上的版本均带有 S7 GRAPH 编程语言。适用于 SIMATIC S7－300、400、C7 和 WinAC，其特点是以图形方式，快速准确地组织和编写 PLC 系统的顺序控制程序。即之前介绍的顺序功能图绘制好之后还需要翻译成梯形图或者功能块图，而这种方法直接用非常近似的图形化编程语言实现，直观明了。

### 1. 顺序控制程序结构

顺序功能图需要 3 个块：调用 S7 GRAPH 的块、S7 GRAPH FB 块（描述顺序控制的块）和 S7 GRAPH FB 的背景数据块。其包含的分支数越多，则执行的时间越长，分支数的多少和 CPU 的类型有关。

### 2. 顺序控制器的工具条含义

顺序控制器工具条的含义如图 6-65 所示。




图 6-65　顺序控制器工具条的含义

### 3. 顺序控制的常用动作指令

如前所述，顺序控制的常用动作为保持型和非保持型两大类型，此外还包括时间调用的方式、块的调用方式等。具体如表 6-8 所示。

**表 6-8　顺序控制的标准动作**

| 动作指令 | 地址标识符 | 注　　释 |
|---|---|---|
| N | I、Q、M、D | 步处于活动状态，地址置为 1，不保持 |
| S | I、Q、M、D | 步处于活动状态，地址置为 1，保持 |
| R | I、Q、M、D | 步处于活动状态，地址置为 0，保持 |
| D | I、Q、M、D | 步处于活动状态并延时时间 T#（const）后，若步仍为活动步则动作为 1，不活动则为 0 |
| | T#（const） | |
| L | I、Q、M、D<br>T#（const） | 步处于活动状态，在脉冲时间 T#（const）内，若步仍为活动步则动作为 1，不活动则为 0 |
| CALL | FB、FC、SFB、SFC | 步处于活动状态，指定块被调用 |

### 4. S7 GRAPH 的步进计数器

S7 GRAPH 的步进计数器如表 6-9 所示。动作中的计数器依赖于事件。计数器可以具有互锁功能。对于具有互锁功能的计数器，只有在互锁条件满足和事件发生时，动作中的计数器才会计数。对于无互锁功能的计数器，则事件发生时，动作中的计数器才会自动计数。

表 6-9　S7 GRAPH 中的步进计数器

| 事　件 | 指令 | 地址标识符 | 数据位 | 注　释 |
|---|---|---|---|---|
| S1、S0、L1、L0、V1、V0、A1、R1 | CS[C] | C | X | 设置:事件一出现[并且自锁条件满足],设定的计数值就装载到该计数器 |
| | | <initial counter value> | — | 初始的计数值 |
| S1、S0、L1、L0、V1、V0、A1、R1 | CU[C] | C | X | 增计数:事件一出现[并且自锁条件满足],计数值增加 1 |
| S1、S0、L1、L0、V1、V0、A1、R1 | CD[C] | C | X | 减计数:事件一出现[并且自锁条件满足],计数值减 1 |
| S1、S0、L1、L0、V1、V0、A1、R1 | CR[C] | C | X | 复位:事件一出现[并且自锁条件满足],计数值复位为 0 |

在表 6-9 中，[ ] 表示可选择带互锁功能，X 表示计数器序号，X：0～999。所有计数功能包括设置计数值，即需要一个初始的计数值，初始的计数值通过下面的类型来设置。

计数的初始值：IW、QW、MW、DBW、DIW 等。

计数器变量：C0～C999。

### 5. S7 GRAPH FB 的基本参数集

在 S7 GRAPH 程序编辑器中，执行菜单命令“Options/Block Settings”，可以在“Compile/Save”选项卡中的“FB Parameters”区中进行选择。其中 S7 GRAPH FB 基本的输入参数如表 6-10 所示。输出参数如表 6-11 所示。

表 6-10　S7 GRAPH FB 的基本输入参数

| 参　数 | 数据类型 | 参数说明 |
|---|---|---|
| EN | BOOL | 使能输入，控制 FB 的执行 |
| OFF_SQ | BOOL | 关闭顺序控制器，使所有步变成不活动步 |
| INIT_SQ | BOOL | 激活初始步，复位顺序控制器 |
| ACK_EF | BOOL | 确认错误和故障，强制切换到下一步 |
| SW_AUTO | BOOL | 切换到自动模式 |
| SW_MAN | BOOL | 切换到手动模式 |
| SW_TAP | BOOL | 切换到单步模式 |
| T_PUSH | BOOL | 在其上升沿时，开始控制 |

表 6-11　S7 GRAPH FB 的基本输出参数

| 参　数 | 数据类型 | 参数说明 |
|---|---|---|
| ENO | BOOL | 使能输出，执行 FB 没有错误则为 1，否则为 0 |
| ERR_FLT | BOOL | 有故障 |
| AUTO_ON | BOOL | 指示自动模式 |
| TAP_ON | BOOL | 指示半自动模式 |
| MAN_ON | BOOL | 指示手动模式 |

### 6. S7 GRAPH 的编程步骤

下面以一个控制实例说明 S7 GRAPH 的编程步骤。

**例 7**　试设计电动机起动控制系统，要求按下按钮 I0.0 一次，第一台电动机 Q0.0 起

动，再次按下按钮，第 2 台电动机 Q0.1 起动，第 3 次按钮按下时，第 3 台电动机 Q0.3 起动，之后，按钮第 4 次按下时，3 台电动机同时停止。之后依次循环。

1）建立项目，插入 S7 程序。

2）点击块，插入新对象/功能块，选择语言为 GRAPH，如图 6-66 所示。




图 6-66　功能块语言选择

3）双击功能块 FB1，打开顺序功能图编辑器，如图 6-67 所示。

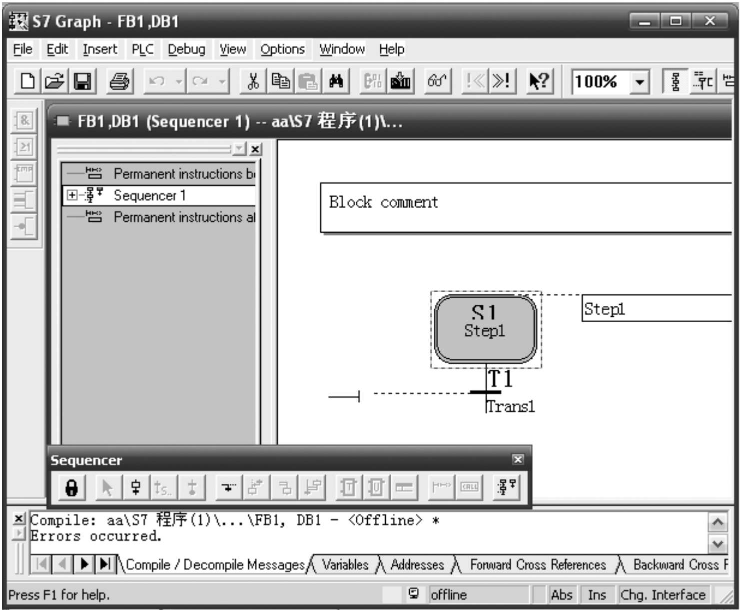



图 6-67　顺序功能编辑器

4）编写顺序控制程序。

① 插入条件时，点击条件处，右键插入需要的逻辑关系（与、或、比较器）（或者点击快捷按钮 ），如图 6-68 所示。如果为一个逻辑条件则简单插入“与”或者是

"或"的关系即可。如果为多个逻辑关系，还可以添加管角、添加反相门，之后输入相应的条件。

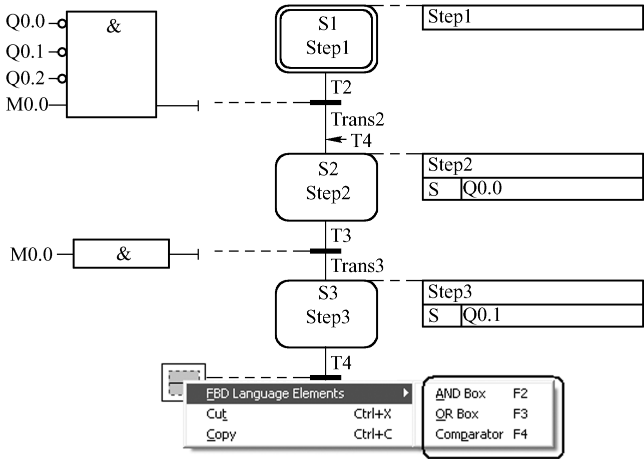



图 6-68　插入条件

② 插入动作时，点击条件处，右键插入动作（Action）（或者点击快捷按钮），如图 6-69 所示。输入所需要的动作。

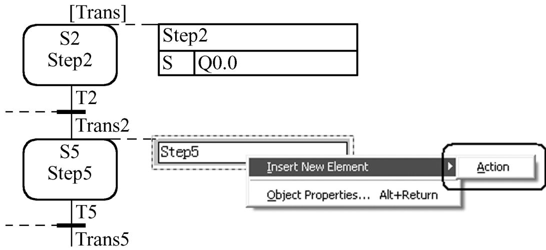



图 6-69　插入动作

③ 插入新的步时，点击最下方的 TransN，右键选择插入新的步（或者点击快捷按钮），如图 6-70 所示。还可以根据需要插入跳转，或者分支。

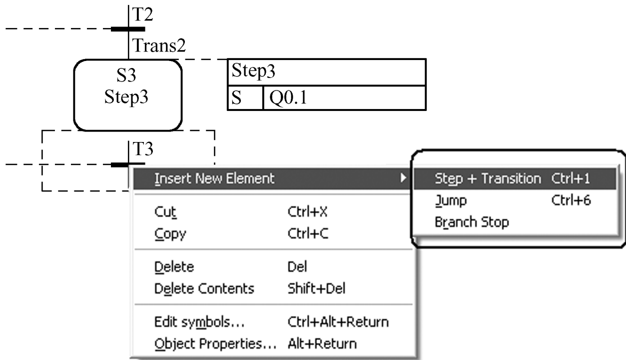



图 6-70　插入新的步

之后根据控制要求完成 GRAPH 程序。如图 6-71 所示为电动机控制 GRAPH 程序。

说明：按照要求，I0.0 每按下一次，则电动机相应起动或者停止，而 GRAPH 程序中却没

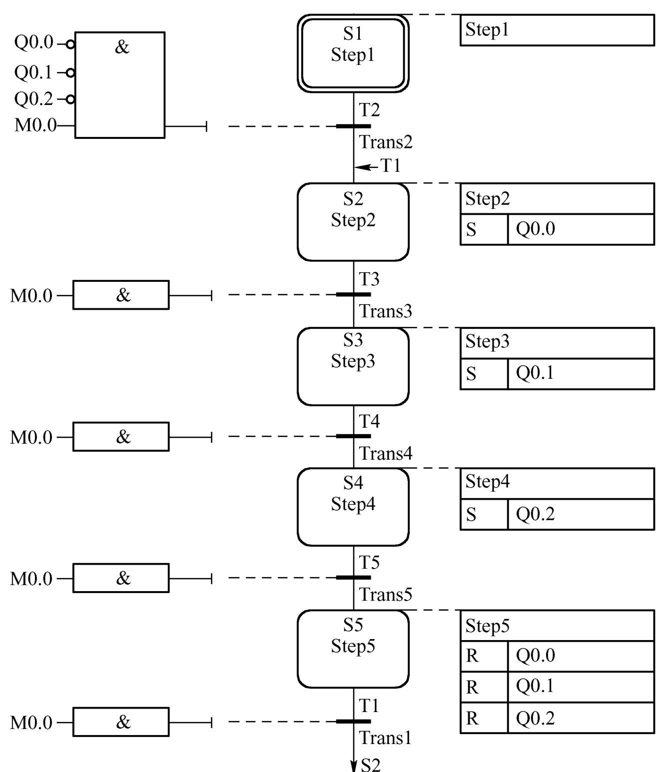



图 6-71　电动机控制 GRAPH 程序

有用 I0.0 作为转换条件，这是因为 I0.0 按下期间，由于转换条件相同，可以使程序运行好几步，这样，I0.0 按下的时间长短不一，运行的步数也不一致，不符合每按一次，运行一步的要求。因此用 M0.0 作为了转换条件，而 M0.0 是在按钮 I0.0 按下后，只导通一个扫描周期（体现在 OB1 中的 Network2），确保 GRAPH 程序每次按下 I0.0 只运行一步。程序中包含动作指令 S 和 R。

5）保存 FB1，经过 FB1 的自动编译后，无错误，则系统自动生成 FC72 和 SFC64 功能。

6）双击 OB1，调用 FB1。OB1 程序如图 6-72 所示。

7）打开模拟仿真器 PLCSIM，CPU 打到 RUN－P 状态，下载所有程序。

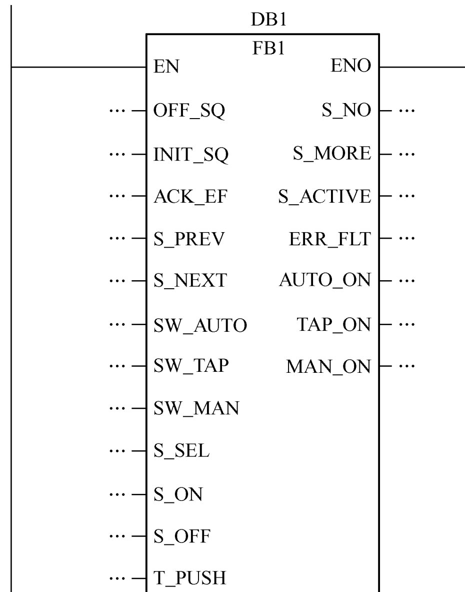
8）在 FB1 功能模块中，点击监控按钮，如图 6-73 所示调试程序。调试时需要仿真 PLC 的配合。

**例 8**　前述的 FESTO MPS 检测站的工作过程为：检测工件的颜色和材料→提升汽缸上升→测量工件的高度。如果测试正确：打开气垫滑动模块→将推出汽缸的活塞杆推出→将推出汽缸的活塞杆缩回→关闭气垫滑动模块→将提升汽缸下降→返回初始状态。如果测试失败：将提升汽缸下降→将推出汽缸的活塞杆推出→将推出汽缸的活塞杆缩回→返回初始位置。其初始状态为：工件在工件托盘中；工作区是空的（旋转臂不在工件托盘处）；提升汽缸在底部；推出汽缸在缩回位置；气垫滑动模块在关闭状态。下面用 GRAPH 语言来实现。

检测站的符号表如图 6-74 所示。

OB1：" 主程序 "

Network 1：顺序控制程序的调用




Network 2：

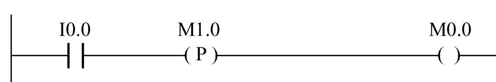



图 6-72　电动机控制 OB1 程序

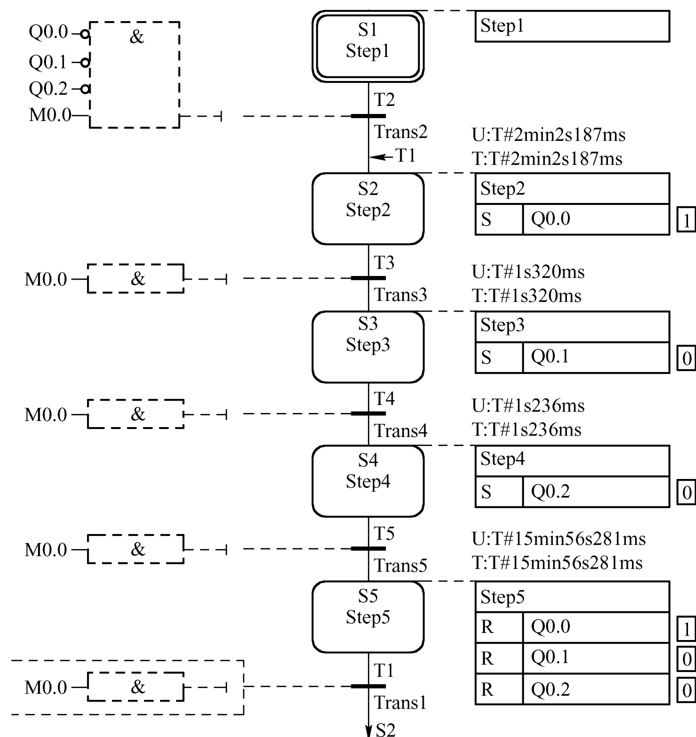



图 6-73　电动机控制 GRAPH 程序监控

02PR_KFA (Symbole) -- mps_control

| | Status | Symbol | Address | Data type | Comment |
|---|---|---|---|---|---|
| 1 | | 1B1 | I 0.4 | BOOL | 提升气缸在高位 |
| 2 | | 1B2 | I 0.5 | BOOL | 提升气缸在低位 |
| 3 | | 1Y1 | Q 0.0 | BOOL | 提升气缸下降 |
| 4 | | 1Y2 | Q 0.1 | BOOL | 提升气缸升起 |
| 5 | | 2B1 | I 0.6 | BOOL | 推出气缸在缩回位置 |
| 6 | | 2Y1 | Q 0.2 | BOOL | 推出气缸的活塞杆推出 |
| 7 | | 3Y1 | Q 0.3 | BOOL | 气垫滑动打开 |
| 8 | | B2 | I 0.1 | BOOL | 测试黑工件 |
| 9 | | B4 | I 0.2 | BOOL | 检测旋转臂是否工件容纳器中,空为1 |
| 10 | | B5 | I 0.3 | BOOL | 工件高度正确 |
| 11 | | IP_F | I 0.7 | BOOL | 下一站准备就绪 |
| 12 | | IP_N_FO | Q 0.7 | BOOL | 本站就绪 |
| 13 | | Part_AV | I 0.0 | BOOL | 工件检测 |
| 14 | | S1 | I 1.0 | BOOL | 起动按钮 |
| 15 | | S2 | I 1.1 | BOOL | 停止按钮 |
| 16 | | S3 | I 1.2 | BOOL | 自动/单步按钮 |
| 17 | | S4 | I 1.3 | BOOL | 复位按钮 |

图 6-74　检测站的符号表

检测站程序中，实现自动运行和复位操作的程序块分别为 FB2 和 FB1，用 GRAPH 编写。其自动运行程序 FB2 如图 6-75 所示。

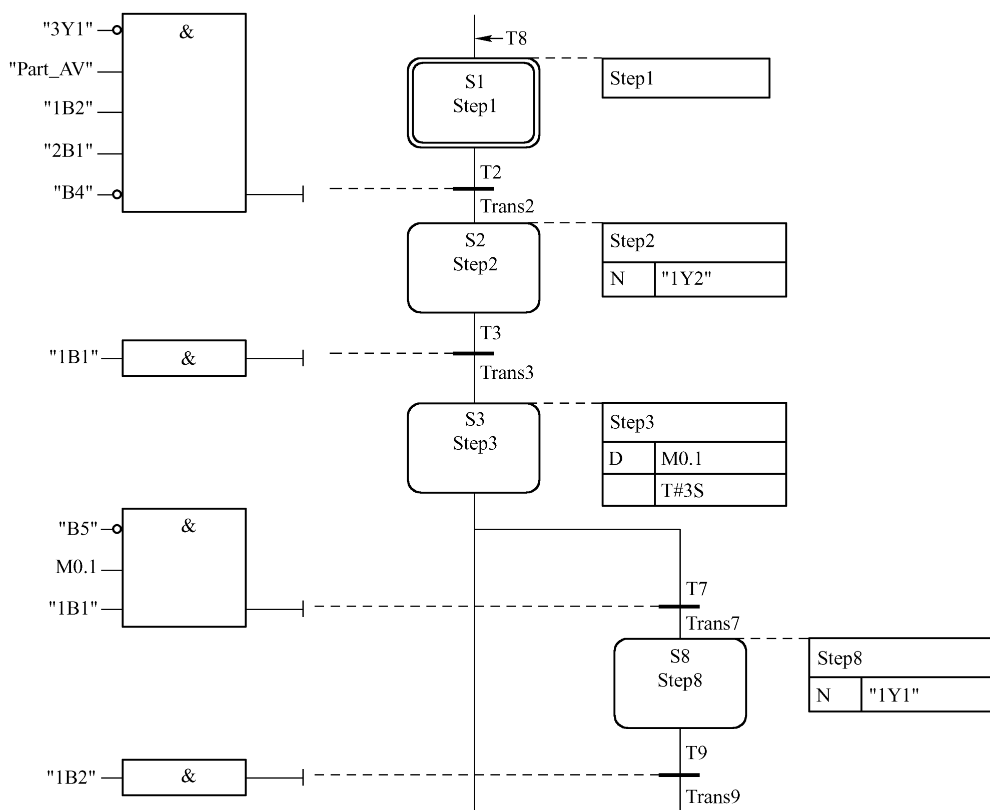



图 6-75　检测站自动运行程序 FB2

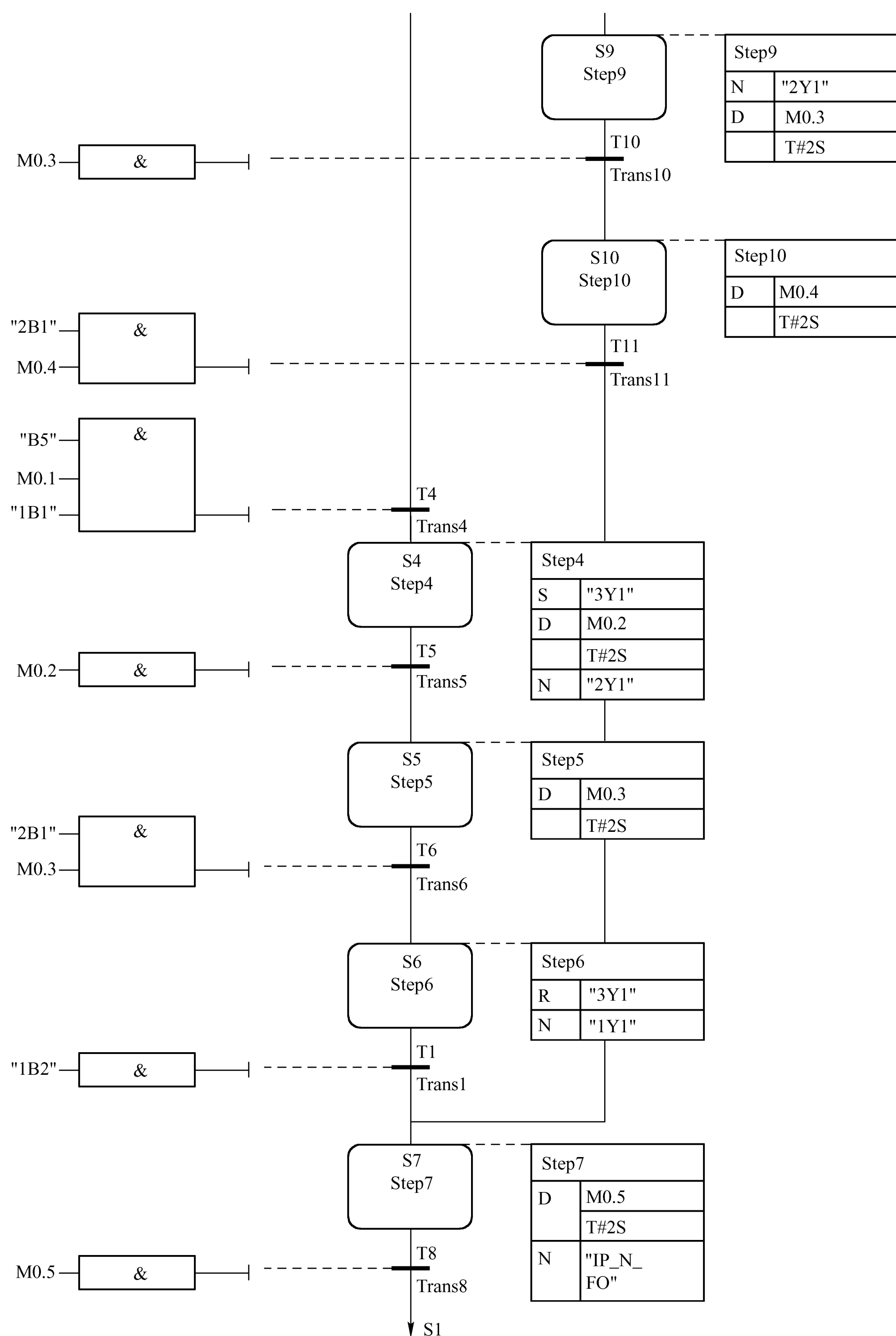



图 6-75　检测站自动运行程序 FB2（续）

说明：GRAPH 程序中有选择序列。并且还包含了延时动作指令。当选择序列和并行序列在分支和合并处难以实现时（没有合适的分支或者合并位置），则加个空步就可以解决。

复位程序如图 6-76 所示。

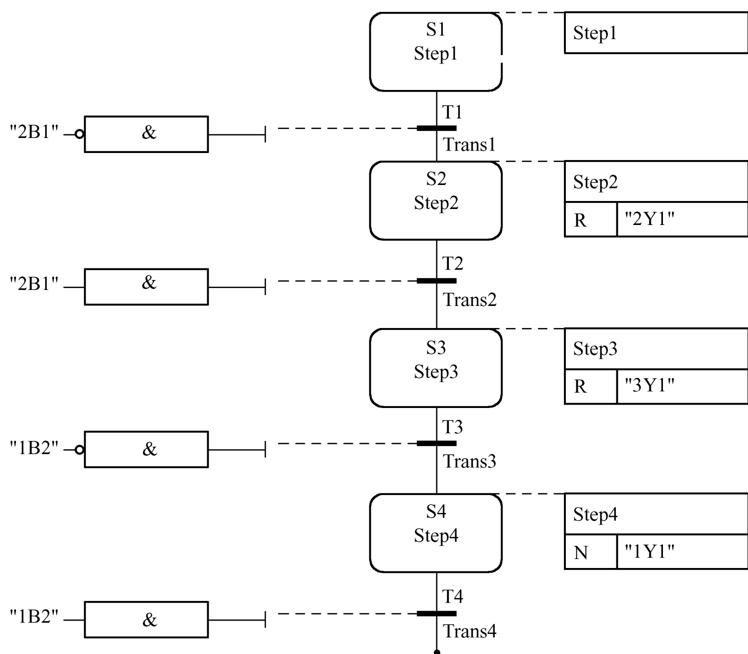



图 6–76　检测站复位程序 FB1

主程序 OB1 如图 6–77 所示。

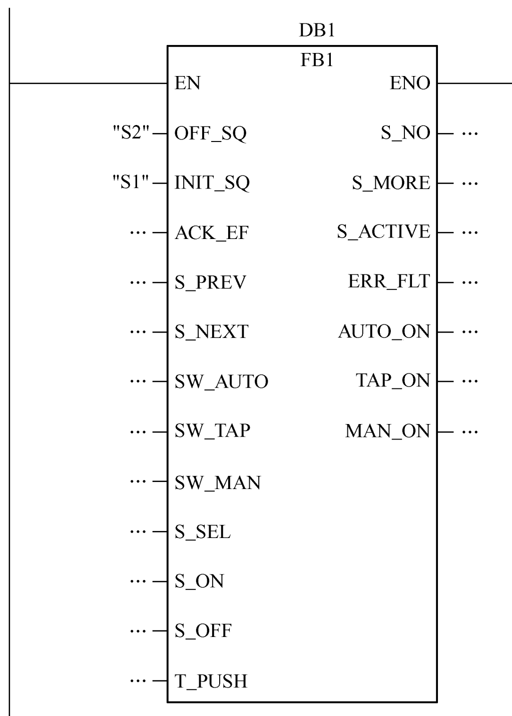



图 6–77　主程序 OB1

Network 2：复位程序

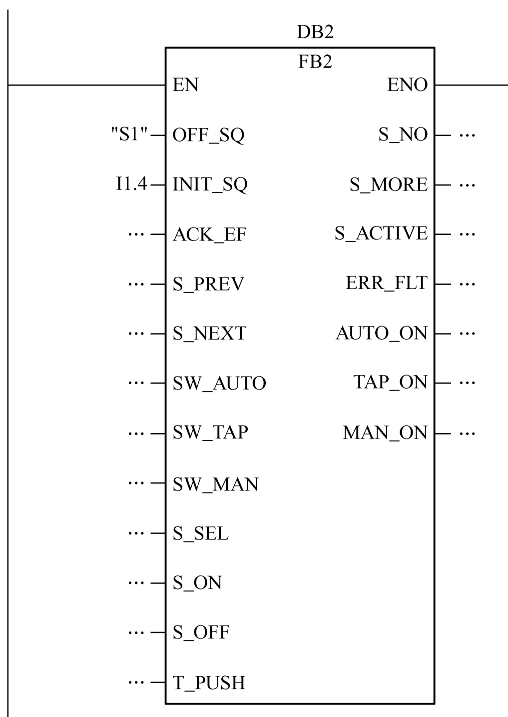



图 6-77　主程序 OB1（续）

S7 GRAPH 以图形化方式清晰表明了控制的过程，功能强大，可以实现多种运行方式；并且其调试方便，便于掌握。

# 6.5　状态图（State Graph）

## 6.5.1　状态图简介

状态图是另外一种有效的编程方法，它与顺序功能图的分析法比较相似，但应用的侧重点不同，顺序功能图侧重于顺序控制过程，而状态图侧重于异步的非顺序过程。状态图能通过状态和状态间的转换来描述设备装置的功能元素或者系统的工作过程。典型的状态图注解图如图 6-78 所示。

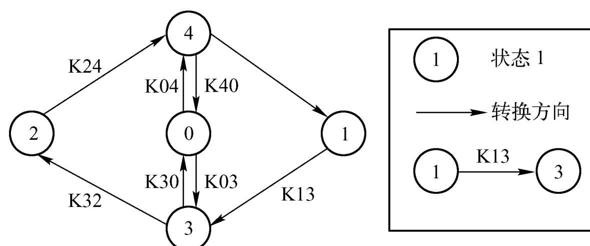



图 6-78　状态图注解图

状态图包含三个要素：状态（圆形表示）、转换条件、转换方向（带箭头的直线）。只要当前状态是在状态 1 位置，同时满足转换条件 K13。图 6-64 中的状态 1 就可以转换到状态 3，其中的箭头表示转换的方向。

状态图的特点是：

1）状态图在某一时刻只能激活一种状态，它可以有分支，也可以有循环。

2）一种状态转换到另一种状态，必须有转换条件。只有在当前状态被激活同时满足转换条件时，才能完成两个状态之间的转换。

3）状态有时用矩形取代圆形，有时两种符号都使用，区别在于：用矩形表示没有运动的状态，用圆形表示有运动的状态。

在 PLC 编程方法中，顺序功能图和状态图可以更完善地描述系统，说明更清晰、简洁，更容易理解、扩展和改进。

顺序功能图和状态图相比较，则状态图有如下特点：

1）允许以不同的方式将控制程序层次化和模块化，这将减少设计的复杂性。

2）允许分支和循环，这一点在顺序功能图中实现会比较复杂。

3）因为建立了相关元件及任务的状态模型，所以更加容易理解。

4）允许包含错误监控。

## 6.5.2 状态图的建立方法及状态图的程序实现

### 1. 基础状态图的建立

建立基础状态图有两种方式。我们可以通过任务描述（功能图或流程图）建立状态图，还可以根据物理装置特点建立功能元素的真实模型状态图（每个执行机构都可以看作是功能元素）。

如图 6-79 所示显示了功能元素（汽缸装置）及流程图。

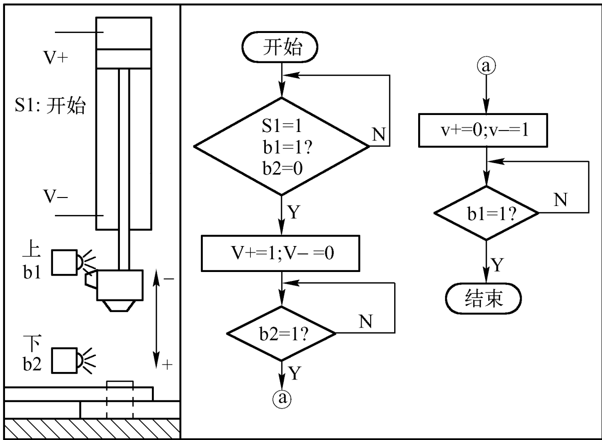



图 6-79　功能元素（汽缸装置）及流程图

用原理图描述的汽缸装置的工作流程为：当活塞在上位置，且碰到行程开关 b1 时，则活塞向下移；当碰到行程开关 b2 时，活塞向上移。通过流程图可以看出汽缸装置有三种基本状态：上位置、向上移、向下移，三种基本状态在满足一定转换条件下构成回环，加上初

始位置（进入回环的入口）共 4 个状态。根据任务描述建立的汽缸装置的状态图如图 6-80 所示。

根据物理装置的特点分析可知：汽缸装置有 4 种基本状态：上位置、下位置、向上移和向下移，加上初始位置共 5 个状态。根据物理装置特点建立的汽缸装置的状态图如图 6-81 所示。

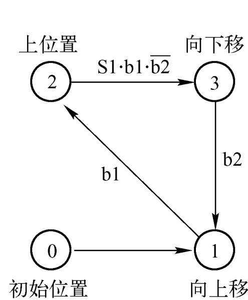



图 6-80　根据任务描述建立的汽缸装置的状态图

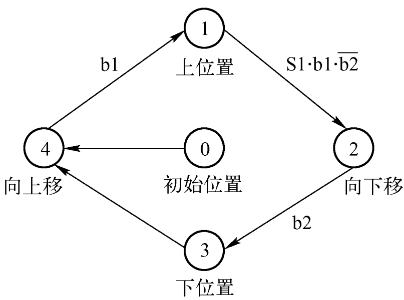



图 6-81　根据物理装置特点建立的汽缸装置的状态图

物理建模的优点是：当对功能元素进行真实行为的建模时，很容易对控制任务进行修改和扩展。在汽缸装置模型中如果要求修改功能为：汽缸挤压活塞，2 s 之后到达下降位置。那么两种方式中究竟哪一种更容易扩展呢？我们又如何扩展呢？这个问题留作思考。

**2. 流出状态图的建立**

如图 6-82 所示为液压钻孔装置，按照图中所示将两个汽缸装置构成一套钻孔单元，使用其中一个汽缸装置夹紧工件，另一个汽缸装置进给钻孔。现在问题是如何建立系统状态图？如何利用状态图完成控制任务？

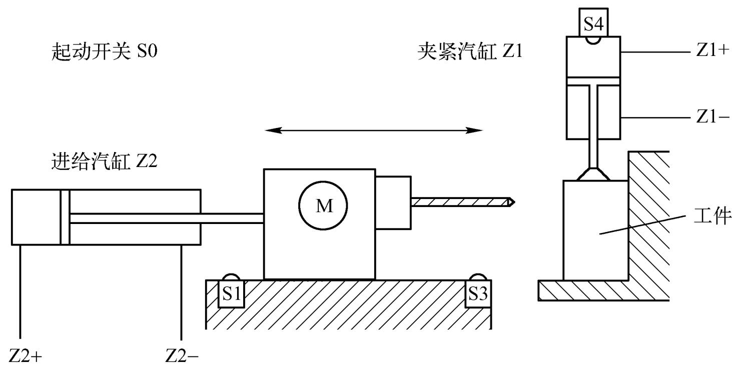



图 6-82　液压钻孔装置

液压钻孔系统包括两个汽缸装置，而我们之前已经建立了汽缸装置的模型，这样，系统包含两个基础状态图，且每个状态图表示一个汽缸装置。因为这两个基本状态图是独立的，所以需要协调（同步）两者以达到期望的钻孔功能，流出状态图就是起这个作用。它可以通过用带点画线的箭头从一个状态图的某一状态转换到另一个状态图的另一状态。只要能够决定状态图之间的转换条件，就可以用流出状态图同步两个基础状态图。

基础状态图的相互同步主要利用了状态图可以分支的特点。借助这一特点可以通过一个流出状态图建立两个（甚至多个）基础状态图的相互连接。流出状态图的主要作用就是同步基础状态图。

如果基础状态图之间没有同步箭头，可以利用流出状态图通过同步箭头和基础状态图相连接。可以这样比拟：如果流出状态图是树根，那么基础状态图就是树干。通过一个流出状态图同样可以同步几个流出状态图，这样可以扩展分支，使之枝繁叶茂。

通过上述方法，可以扩展分支，同时可以将控制结构模块化，这样可以减少任务的复杂性。

在上述的液压钻孔装置中，可以通过一个流出状态图同步两个基础状态图。钻孔装置的状态图简图如图 6–83 所示。

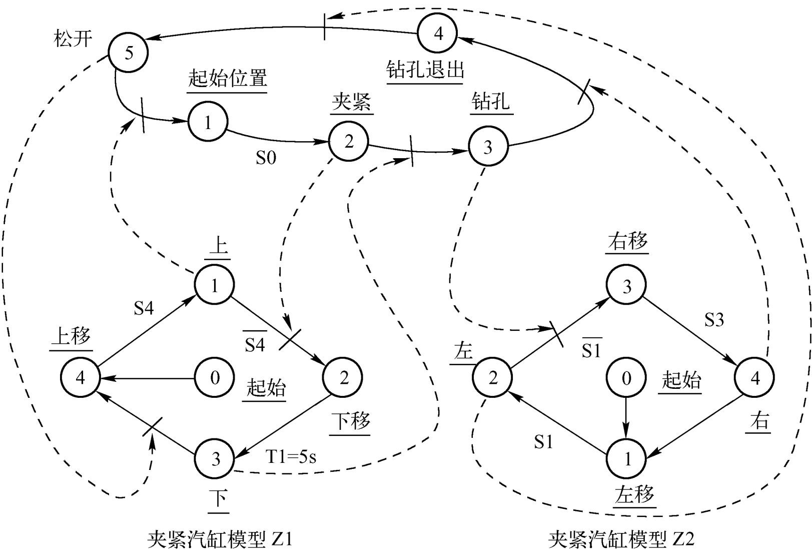



图 6–83　钻孔装置的状态图简图

分析：从图中可以看出，为完成系统钻孔任务，流出状态图包括 5 个状态：起始位置、夹紧、钻孔、钻孔退出和松开。在流出状态图的“起始位置”，夹紧汽缸在上位，（此时进给汽缸在左边），有虚线箭头从夹紧汽缸基础状态图指向流出状态图，表示夹紧汽缸的位置。在起动开关（S0）闭合时，流出状态图到“夹紧”状态，此时夹紧活塞下移，可以看到虚线箭头从流出状态图指向夹紧基础状态图，满足夹紧活塞从上位向下移的条件。夹紧活塞至下位时，延时 5s，确保夹紧后，满足从夹紧到钻孔的条件，流出状态图处于“钻孔”状态，此时电动机开始工作，对工件钻孔。同样有虚线箭头从进给汽缸模型指向流出状态图。以此类推，这样流出状态图将基础状态图夹紧状态图和进给状态图进行了同步。与此同时，夹紧汽缸和进给汽缸这两个基础状态图之间没有联系，并独立工作。

**3. 系统状态图的建立**

在分析问题时，运用了流出状态图和基础状态图，那么在编程时，如何方便地利用状态图呢？我们知道在顺序功能图中有：转换条件、步以及动作三个要素，同样在状态图中，也需要完善这些要素。在上述的钻孔装置中，由于流出状态图包含了夹紧汽缸模型和进给汽缸

模型的动作，所以在进行系统设计时，只需要进行流出状态图的完善即可（将夹紧汽缸和进给汽缸的动作包含在其中）。完善后（包括转换条件、对应动作）的钻孔装置的状态图如图 6-84 所示。其中状态图中的动作用 S 表示置位，R 表示复位，Start Tn 表示起动定时器 Tn。

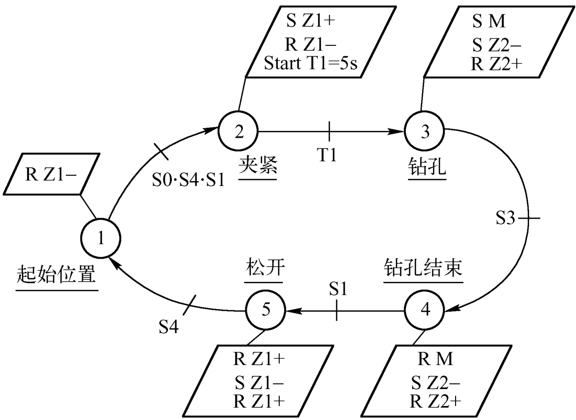



图 6-84　完善后的钻孔装置状态图

分析：由图中可以看出，流出状态图中添加了转换条件，同时还添加了每个状态所执行的动作。比如状态 1（起始位置）表示电动机是停止状态（R M），夹紧汽缸在上位（R Z1 +），进给汽缸在左边（R Z2 +）；状态 2（夹紧）表示夹紧汽缸在下位（S Z1 +，R Z1 −），为保证夹紧，进行了延时，其余状态以此类推。注意：在状态 3 中，为了钻孔，电动机带动钻头转动（S M），在钻孔结束后的状态 4 中电动机停止（R M）。

状态图同样适合错误监控，可以通过出错状态和转换延伸系统控制图，或者扩展另一个状态图来执行监控。

**4. 状态图的程序实现方法**

状态图的实现最好在功能块 FB 中进行。因为功能块带背景数据块，它可以存储状态图中的激活状态数。当然我们也可以用一个全局存储器字来存储激活的状态数。为了防止因偶然因素而覆盖状态数，我们选择背景数据块则更为保险，这就是使用功能块 FB 实现状态图的原因。

由于状态图适合于异步非顺序过程的描述，其中可能有许多分支和跳转，所以用语句表编程更为适合。

利用语句表建立钻孔装置状态图的程序为：

```
FB4:钻孔装置 STL 程序
Network1:循环执行部分
    L    S5T#5S          //为夹紧工件,定时器 T1 延时 5s
    A    M10.0
    SD   T1
Network2:激活状态存储在 DB 块中,如果为 n 跳转到 Cn 标签处(为转换状态使用)
    L    #ZNr            //将激活状态数从数据块中调出,#ZNr 为定义的静态局部变量
    JL   Err1
```

```
      JU    C0                //如果为 0,跳转到 C0
      JU    C1                //如果为 1,跳转到 C1
      JU    C2                //如果为 2,跳转到 C2
      JU    C3                //如果为 3,跳转到 C3
      JU    C4                //如果为 4,跳转到 C4
      JU    C5                //如果为 5,跳转到 C5
      Err1:BEU                //超出范围,则停止
Network3:转换部分
      C0:  JU       S1
           BEU
      C1:  A        "S0"
           A        "S4"
           A        "S1"
           JC       S2    //从状态 1 转换到状态 2
           BEU
      C2:  A        T1
           JC       S3    //从状态 2 转换到状态 3
           BEU
      C3:  A        "S3"
           JC       S4    //从状态 3 转换到状态 4
           BEU
      C4:  A        "S1"
           JC       S5    //从状态 4 转换到状态 5
           BEU
      C5:  A        "S4"
           JC       S1    //从状态 5 转换到状态 1
           BEU
Network4:存储新状态,并跳转到新状态标签 Bn(执行动作)
      S1:  L        1
           JU       CS
      S2:  L        2
           JU       CS
      S3:  L        3
           JU       CS
      S4:  L        4
           JU       CS
      S5:  L        5
           JU       CS
           BEU
      CS:  T        #ZNr  //存储状态数
           JL       Fauz
           JU       B0    //如果为 0,则跳转到 B0
           JU       B1    //如果为 1,则跳转到 B1
```

```
        JU        B2     //如果为 2,则跳转到 B2
        JU        B3     //如果为 3,则跳转到 B3
        JU        B4     //如果为 4,则跳转到 B4
        JU        B5     //如果为 5,则跳转到 B5
        Fauz:     BEU    //超出范围,则停止
Network5:状态对应的动作
    B0: BEU
    B1: SET
        R         "Z1 -"//夹紧活塞在上位,则停止夹紧气缸的运动
        R         M10. 0
        BEU
    B2: SET
        S         "Z1 +"//夹紧气缸夹紧工件
        R         "Z1 -"
        S         M10.0 //起动定时器 T1
        BEU
    B3: SET
        S         "Z2 +"//工件进给
        S         "M"   //起动电动机钻孔
        R         "Z2 -"
        BEU
    B4: SET
        R         "M"   //停止电动机
        S         "Z2 -"//进给气缸退后
        R         "Z2 +"
        BEU
    B5: SET
        R         "Z2 -"//当进给活塞在左边,则停止进给气缸的运动
        R         "Z1 +"//夹紧气缸松开
        S         "Z1 -"
        BEU
```

程序结构分析：程序有 5 个段落，Network1 是处理循环执行的程序；Network2 的作用是从数据块中调出激活状态数，并跳转到相应的状态位置（这里涉及多分支跳转指令）；Network3 为转换部分，其作用是根据状态图的转换条件确定状态之间的转换；Network4 的作用是存储状态数，存储的状态数是为了停止后状态的恢复；Network5 是状态所对应的动作。程序的 5 个段落形成状态图的结构。其中 Network3 和 Network5 是程序的核心，表示状态的转换及执行的动作。当用户选择好状态图的状态（比如是 5 个状态）时，不必对程序结构做大的修改，只需复制 FB 块中的 Network2 和 Network4 到另一个功能块 FB 中，并作适当的扩充，同时根据不同的条件和执行的动作填充 Network3 和 Network5 即可。

说明：当 PLC 起动时，我们开始执行初始状态 0。在 Network2 中跳转到 Network3 的标号 C0 处，标号为 C0 的段落包含状态 0 的转换和起动的初始状态。在本例中我们无需作任

何事，而只需无条件跳转到 Network4 的标号 S1 处，S1 表示状态 1。在 Network4 中将新状态 1 存储到 DB 块的参数#ZNr 中，因此我们可以在 Network5 中执行状态 1 的动作：停止电动机，夹紧气缸向上，进给气缸向左。在下一次 PLC 循环中 Network3 只要检查这些转换，使之从激活的状态 1 转换到其他状态。如果没有状态是激活的，则功能块 FB 不执行；如果一个状态激活且转换条件满足时，则跳转到 Network4 中，在 Network4 中新的状态数被存储。有了新的状态，我们可以在 Network5 中执行新状态的动作。

值得注意的是在状态图功能块 FB 的执行过程中，可能得不到 PLC 的监控信息。因为在许多 PLC 的循环状态情况下，如果没有任何状态发生变化，或者所有转换条件导致没有新的实际状态激活时，FB 将结束过程。在 Network3 中也只有标号数在 C0～C5 的信息可以监控。

这似乎是一个缺点，因为难以发现合适的位置来监控信息。而实际上这是一个优点。我们知道实际状态存储在 DB 块的参数#ZNr 中，因此知道程序中监控信息的位置。当实际激活状态不能转换到另一个状态时，我们可以关注这个信息，了解为什么相关转换条件不能满足，同样可以看出是哪个信号导致程序执行出错。

在其他情况下，有时也会发生转换条件一直满足状态始终不变的情况，这可能是因为几个状态之间的快速转换。问题可能是由于这一个转换条件在几个状态转换条件中都包含。

### 6.5.3 状态图应用实践

**例 9** 前述的果汁浓缩系统，试通过状态图编写语句表程序。

（1）建立系统状态图

通过系统分析可知：系统需要有 5 个输出元件（功能元素），分别为阀门 V1、V2、电动机 M、加热器 H 和指示灯 L。这些元件需要位信号控制，因此控制的复杂性不高，相应的基础状态图不需要以原始的物理建模方式来建立。

首先，我们将功能元素分组，形成几个功能单元。例如为果汁罐定义的基础状态图包括 V1、V2、M、N0、N1、S 和 L。另一个状态图代表加热器 H 和温度检测 ST。这样可以根据元素的结构并考虑系统的分支和改进，对系统进行分析。

其次，我们需要决定果汁罐基础状态图的状态，这一点在控制中十分重要。其中必须考虑的重要状态为“开始位置”、“罐入果汁”、“半满”、“罐满”、“排空”5 个状态。“开始位置”状态指果汁罐是空的状态，并且两个阀门都是关的，指示灯 L 是亮的，同时电动机 M 没有运行。“罐入果汁”状态指阀门 V1 打开，指示灯 L 灭。“半满”状态指果汁高度超过 N0，此时要起动电动机 M。第四个状态为“罐满”，当果汁高度达到 N1 时，我们必须采取行动：关闭阀门 V1，“排空”状态指果汁高度经加热后低于 N0，浓缩完成。此时必须关闭电动机 M，打开阀门 V2，并起动定时器 T1，延时 30s 后，将果汁排空。

另一个加热器的基础状态图中加热器 H 需要状态“停止”、“使能”和“激活"。在“使能”的状态下，加热器满足温度条件，则“激活”。

然后，我们必须定义一个流出状态图来同步两个基础状态图，但是发现这个流出状态图的功能在果汁罐的基础状态图中已经描述，所以我们只要将果汁罐的基础状态图延伸来同步加热基础状态图。在果汁灌入高度达到 N0 时，使能加热器，此后当温度变化时，加热器 H 不断的处于激活和使能状态。当加热到一定程度，果汁高度低于 N0 时，停止加热器，果汁浓缩状态图的初步结构如图 6-85 所示。在这个简单例子中相互同步比建立另一个新的流出

状态图更容易。

最后，我们还必须对转换和动作做进一步的延伸，完善之后的果汁浓缩的状态图如图 6-86 所示。

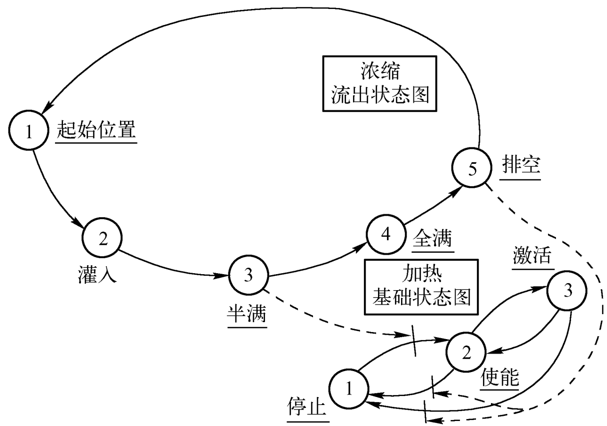



图 6-85　果汁浓缩状态图的初步结构

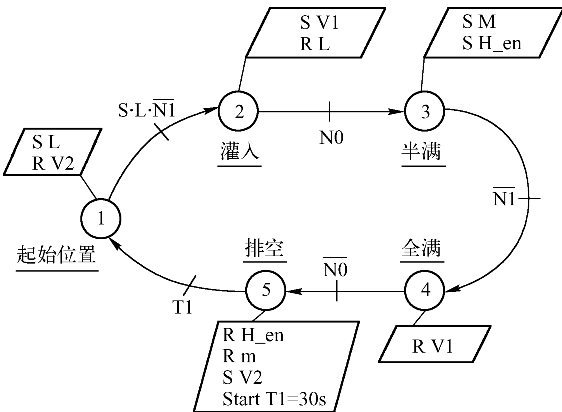



图 6-86　完善之后的果汁浓缩的状态图

（2）语句表的实现程序

语句表的实现程序如下：

```
FB3:果汁浓缩系统的语句表程序
Network1:循环执行部分
    L      S5T#30S          //为了排空果汁,延时 30s
    A      M10.0
    SD     T1

    A      "ST"             //在温度范围内加热器加热,浓缩果汁
    A      "H_en"
    =      "H"
Network2:激活状态存储在 DB 块中,如果为 n 跳转到 Cn 标签处(为转换状态使用)
    L      #ZNr             //从数据块中调出激活状态数
```

```
    JL    Err1
    JU    C0                  //如果为 0,跳转到 C0
    JU    C1                  //如果为 1,跳转到 C1
    JU    C2                  //如果为 2,跳转到 C2
    JU    C3                  //如果为 3,跳转到 C3
    JU    C4                  //如果为 4,跳转到 C4
    JU    C5                  //如果为 5,跳转到 C5
    Err1:BEU                  //如果数字超出范围,则停止
Network3:转换部分
    C0:  JU        S1      //C0 直接转换到 S1
         BEU
    C1:  A         "S"     //在满足转换条件下,C1 转换到 S2
         A         "L"
         AN        "N1"
         JC        S2
         BEU
    C2:  A         "N0"    //在满足转换条件下,C2 转换到 S3
         JC        S3
         BEU
    C3:  AN        "N1"    //在满足转换条件下,C3 转换到 S4
         JC        S4
         BEU
    C4:  AN        "N0"    //在满足转换条件下,C4 转换到 S5
         JC        S5
         BEU
    C5:  A         T1      //在满足转换条件下,C5 转换到 S1
         JC        S1
         BEU
Network4:存储新状态,并跳转到新状态标签 Bn(为指行动作使用)
    S1:  L         1
         JU        CS
    S2:  L         2
         JU        CS
    S3:  L         3
         JU        CS
    S4:  L         4
         JU        CS
    S5:  L         5
         JU        CS
         BEU
    CS:  T         #ZNr    //将当前状态数存入数据块中
         JL        Fauz
         JU        B0      //如果为 0,则跳转到 B0
```

```
        JU      B1      //如果为 1,则跳转到 B1
        JU      B2      //如果为 2,则跳转到 B2
        JU      B3      //如果为 3,则跳转到 B3
        JU      B4      //如果为 4,则跳转到 B4
        JU      B5      //如果为 5,则跳转到 B5
   Fauz:BEU             //范围超出,则停止
Network5:动作
   B0:  BEU
   B1:SET
        S       "L"     //点亮指示灯
        R       M10.0
        R       "V2"
        BEU
   B2:SET
        S       "V1"    //打开灌入阀 V1,灌入果汁
        R       "L"     //熄灭指示灯
        BEU
   B3:  SET
        S       "M"     //起动搅拌电动机
        S       "H_en"//使能加热器
        BEU
   B4:  SET
        R       "V1"    //关闭阀 V1(当果汁高度达到 N1)
        BEU
   B5:  SET
        S       M10.0 //延时排空果汁
        R       "M"     //关闭电动机
        R       "H_en"//关闭加热器使能
        S       "V2"    //打开排空阀 V2
        BEU
```

果汁浓缩系统和钻孔装置的流程图比较相似，状态从状态 1 循环执行到状态 5，程序没有多少分支，具有顺序控制的特点，并没有体现状态图的异步非顺序的功能。下面我们将以保险箱开启的例子说明状态图的这个特点。

**例 10** 电子安全保险箱开启系统如图 6-87 所示。

保险箱装置的开启控制要求如下：

1）通过按钮 I0.2 的上升沿，键入三个数字，且通过 MW200 存储在 MW40、MW42 和 MW44 中。

2）当按下“打开”按钮 I0.3 时，三个数字与存储在 MW20、MW22 和 MW24 中的代码数字相比较。

3）如果代码数字和输入数字相同，门开启，否则延时 10min。

4）当按下“关闭”按钮 I0.4 时，关闭保险箱门，同时起动新一轮的运行。

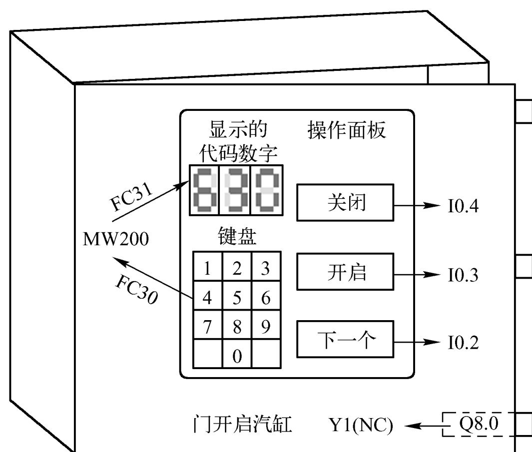



图 6-87　电子安全保险箱开启系统示意图

控制任务：

1）运用经验法，以语句表程序实现系统要求。

2）建立状态图模型，以语句表实现之。

3）比较经验法和状态图法。

首先建立保险箱系统的变量表，如表 6-12 所示。

**表 6-12　保险箱系统的变量表**

| PLC 地址 | 变　量　名 |
|---|---|
| I0. 2 | 输入下一个数字的按钮 |
| I0. 3 | 开启保险箱按钮 |
| I0. 4 | 关闭保险箱按钮 |
| Q8. 0 | 打开保险箱 |
| MW40 | 输入数字 1 |
| MW42 | 输入数字 2 |
| MW44 | 输入数字 3 |
| MW20 | 保险密码 1 |
| MW22 | 保险密码 2 |
| MW24 | 保险密码 3 |

1）运用经验法，以语句表实现系统。现在我们不建立任何中间模型，直接利用语句表来执行控制任务。

程序如下：

FC31：开保险柜，要打开保险柜必须在键盘上输入 3 个数字，它们分别存储在 MW40、MW42 和 MW44 中，正确的代码存储在 MW20、MW22 和 MW24 中。

```
Network1:当输入为错误数字时,阻止开启 10min
      L     S5T#10M
            A           M10.0
            SE          T1
```

```
        R          M10.0 //设置定时器 T1,定时时间为 10min

        L          T1
        A          T1
        BEC
Network2:为下一个输入数字按钮产生边缘检测标志 M0.2
        A          I0.2
        FP         M0.1
        =          M0.2 //产生边缘检测标志 M0.2
Network3:I0.4 初始化开关机构并关闭保险柜
        A          I0.4
        S          M1.0 //初始化,设定第一个数字允许输入标志 M1.0
        R          M2.0
        R          M3.0
        R          M4.0
        R          Q8.0 //关闭保险箱
Network4:存储输入的数字
        A          M0.2
        A          M1.0
        JC         STO1

        A          M0.2
        A          M2.0
        JC         STO2

        A          M0.2
        A          M3.0
        JC         STO3
Network5:密码正确,打开保险柜
        A          I0.3
        A          M4.0 //在密码正确(M4.0 =1)的条件下,开启保险箱
        JC         OPEN

        AN         I0.3
        BEC
Network6:密码不正确,阻止开门 10min
  BLCK: SET
        S          M10.0 //密码不正确,起动定时器延时 10min
        S          M1.0
        R          M2.0
        R          M3.0
        R          M4.0
  BEU
```

Network7:存储输入数字到相应的存储器中

```
STO1: L        MW200
      T        MW40 //输入的第一个数字存储到 MW40 中
      SET
      R        M1.0
      S        M2.0 //设置第二个数字允许输入标志 M2.0
      BEU

STO2: L        MW200
      T        MW42 //输入的第二个数字存储到 MW42 中
      SET
      R        M2.0
      S        M3.0 //设置第三个数字允许输入标志 M2.0
      BEU

STO3: L  MW200
      T        MW44 //输入的第三个数字存储到 MW44 中
      SET
      R        M3.0
      S        M4.0
BEU
```

Network8:比较,如果数字不正确,则跳转到 BLCK,正确则开启

```
OPEN: L  MW20
      L        MW40
      < >I
      JC       BLCK //输入数字 1 不正确

      L        MW22
      L        MW42
      < >I
      JC       BLCK //输入数字 2 不正确

      L        MW24
      L        MW44
      < >I
      JC       BLCK //输入数字 3 不正确

      SET
      S        Q8.0 //数字正确,则打开保险箱
      BE
```

2）建立系统的状态图，根据状态图利用语句表实现。首先分析只有一个保险密码的情况，一个密码的保险箱系统的状态图如图 6-88 所示。

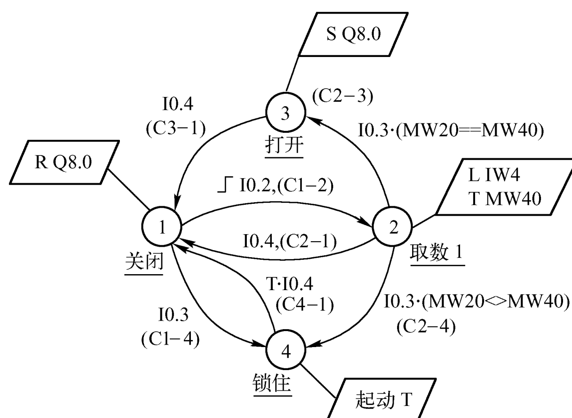



图 6−88　一个密码的保险箱系统的状态图

说明：图中系统有 4 个状态：状态 1 “关闭”、状态 2 “取数 1”、状态 3 “打开” 和状态 4 “锁住”。当“取数 1”值与设置值相同时，状态从“取数 1”转换到“打开”，图中标号为（C2−3）；当取数值与设定值不相同时，状态从“取数 1”转换到“锁住”（C2−4）；当关闭按钮 I0.4 按下时，状态从“取数 1”转换到“关闭”状态（C2−1）。在输入错误时，“锁住”状态延时 10min 后，状态转换到“关闭”保险箱（C4−1）；在没有任何数字输入时，按下打开按钮 I0.3，则状态从“关闭”转换到“锁住”状态（C1−4）；按下关闭按钮 I0.4，则状态从“打开”到“关闭”状态（C3−1）；同样按下取数按钮 I0.2，则状态从“关闭”到“取数 1”状态（C1−2）。

图中状态之间的转换交叉进行，充分体现了非顺序的特点。在此基础上，分析系统的语句表程序的结构比较有意义。

语句表程序结构如下：

如图 6−89 所示为数据块中调出激活状态的流程图（如果是程序刚开启时，则状态值为 0，若系统出现故障，则为故障时存储的状态值），从图中可以看出：程序首先从数据块中装载状态数，根据状态值跳转到 C0 ~ C4。这部分相当于初始化状态。

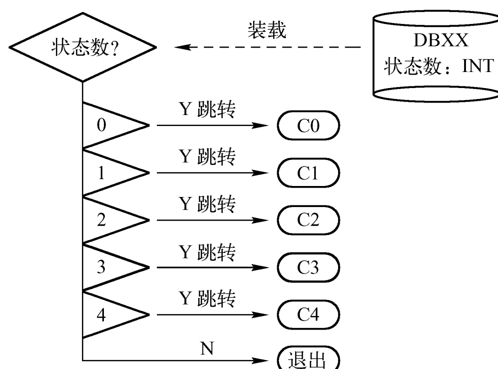



图 6−89　数据块中调出激活状态的流程图

如图 6−90 所示为根据条件进行转换的流程图。例如图中状态 C1 有两个转换方向，即状态 2（C1−2）和状态 4（C1−4），在满足相应转换条件下，分别跳转到 S2 和 S4。这部分相当于根据条件设置转换。

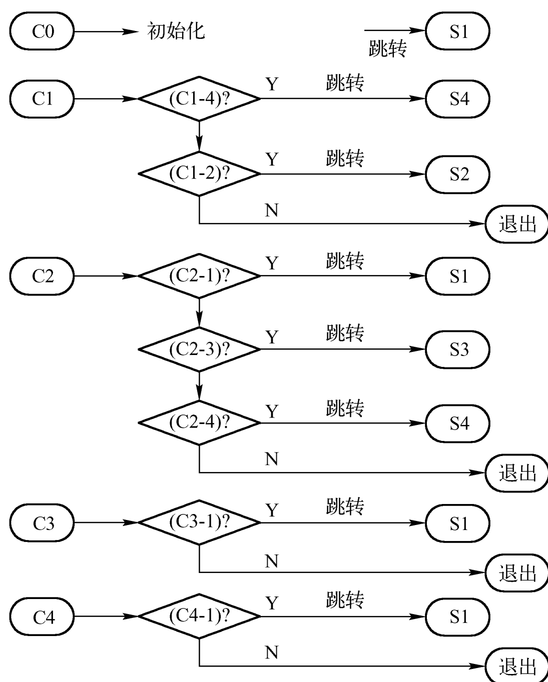



图 6-90　转换的流程图

如图 6-91 所示为故障时存储当前状态并跳转到执行动作状态的流程图。图中 S1 装载数字 1（语句表：L1），并存储于 DB 数据块中（前述的图 6-89 是从 DB 数据块中提取数值），同时跳转到动作状态 B1 处。其余以此类推。注意：存储当前状态的目的主要是当系统由于故障等原因停止运行时，能够重新从程序断点处的状态开始工作。

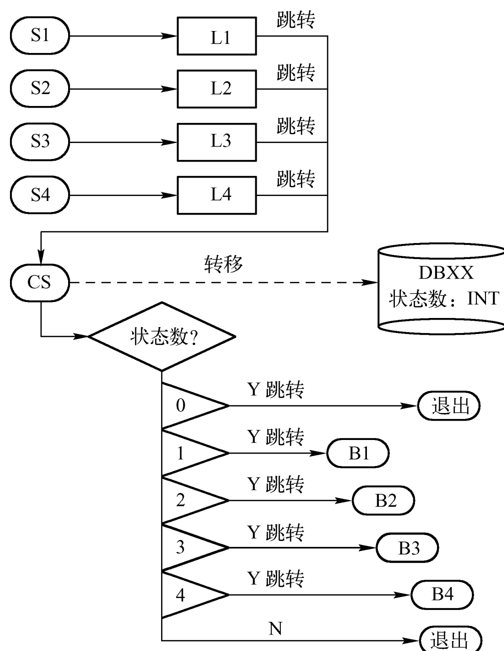



图 6-91　存储当前状态的流程图

如图 6-92 所示为状态执行动作的流程图。图中 B1 表示的动作为关闭保险箱，其余雷同。

如果状态超出正确的范围，则有相应的处理方式。具体表现在程序中的出错处理。

从语句表程序的流程图分析框架中，我们已经比较清楚地了解了状态图的转换及相应的动作实现过程，在此基础上编程将相对简单。

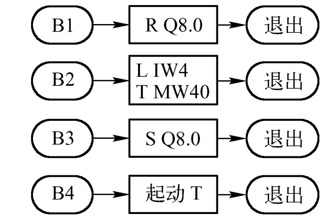


图 6-92　状态执行动作的流程图

语句表实现程序如下：

```
Network1:循环执行部分
    L      S5T#10M           //设置定时器 T1,定时时间 10min
    A      M10.0
    SD     T1
    R      M10.0
    L      T1

    A      I0.2              //建立边缘检测标志 M0.2
    FP     M0.1
    =      M0.2
Network2:从 DB 块中装载激活状态数,如果为 n 跳转到 Cn 标签处(为转换状态使用)
    L      #ZNr              //从数据块中调出激活状态数
    JL     Err1
    JU     C0
    JU     C1
    JU     C2
    JU     C3
    JU     C4
Err1:BEU
Network3:状态转换部分
    C0:    JU      S1
           BEU
    C1:    A       I0.3
           JC      S4       //(C1-4)(表示状态从 1 转换到状态 4)
           A       M0.2     //(C1-2)
           JC      S2
           BEU
    C2:    L       MW20     //(C2-3)
           L       MW40
           ==I
           =       M0.3

           A       M0.3
           A       I0.3
```

```
            JC      S3

            AN      M0.3     //(C2-4)
            A       I0.3
            JC      S4
            A       I0.4     //(C2-1)
            JC      S1
            BEU
    C3:     A       I0.4     //(C3-1)
            JC      S1
            BEU
    C4:     AN      T5       //(C4-1)
            A       I0.4
            JC      S1
            BEU
Network4:存储新状态,并跳转到新状态标签 Bn(为执行动作使用)
    S1:     L       1
            JU      CS
    S2:     L       2
            JU      CS
    S3:     L       3
            JU      CS
    S4:     L       4
            JU      CS
            BEU
    CS:     T       #ZNr     //将当前状态数存入数据块中
            JL      Fauz
            JU      B0
            JU      B1
            JU      B2
            JU      B3
            JU      B4
Fauz: BEU
Network5:动作
    B0:     BEU
    B1:     SET
            R       Q8.0     //关闭保险箱
            BEU
    B2:     L       IW4      //取数
            T       MW40
            BEU
    B3:     SET
            S       Q8.0     //打开保险箱
```

```
         BEU
B4:      SET
         S        M10.0   //起动定时器
         BEU
```

利用上述的分析，可以了解各状态之间的转换情况，这样对于分析有三个保险密码的情况十分有益。如图 6-93 所示为有三个密码的保险箱系统状态图。

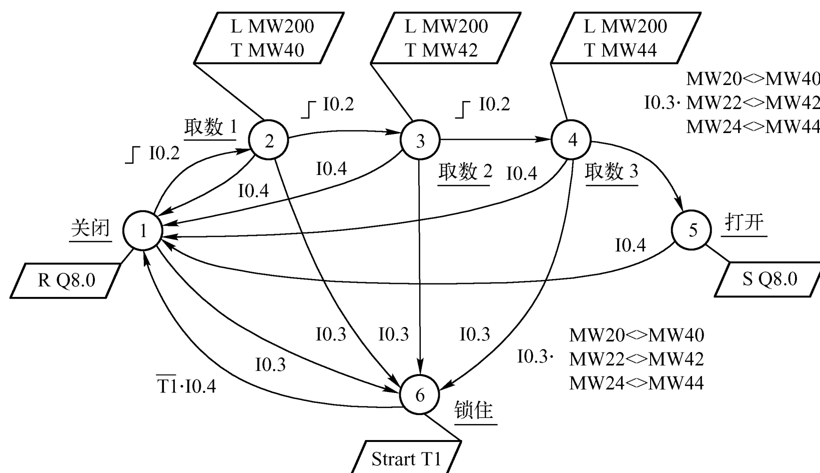



图 6-93　有三个密码的保险箱系统状态图

按照状态图编写的语句表实现程序如下：

```
FB31:保险箱开启程序
Network1:循环执行部分
     L      S5T#10M            //延时 10min
     A      M        10.0
     SE     T        1
     R      M        10.0
     L      T        1

     A      I0.2               //建立边缘检测 M0.2
     FP     M0.1
     =      M0.2
Network2:激活状态存储在 DB 块中,如果为 n 跳转到 Cn 标签处(为转换状态使用)
     L      #ZNr               //从数据块中调出激活状态数
     JL     Err1
     JU     C0
     JU     C1
     JU     C2
     JU     C3
     JU     C4
```

```
      JU    C5
      JU    C6
Err1:BEU
Network3:转换部分
          C0:JU    S1
            BEU
          C1: A    I0.3
            JC     S6          //C1-6
            A      M0.2        //C1-2
            JC     S2
            BEU
          C2:A     I0.4
            JC     S1          //C2-1
            A      M0.2        //C2-3
            JC     S3
            A      I0.3
            JC     S6          //C2-6
            BEU
          C3:A     I0.4
        JC  S1         //C3-1
            A      M0.2        //C3-4
            JC     S4
          A        I0.3
            JC     S6          //C3-6
            BEU
          C4:A     I0.4
            JC     S1          //C4-1
            L      MW20        //C4-5
            L      MW40
            ==I
            =      M0.3
            L      MW22
            L      MW42
            ==I
            =      M0.4
            L      MW24
            L      MW44
            ==I
            =      M0.5
            A      M0.5
            A      M0.4
            A      M0.3
            A      I0.3
```

```
      JC      S5
      L       MW20       //C4-6
      L       MW40
      < >I
      =       M1.3
      L       MW22
      L       MW42
      < >I
      =       M1.4
      L       MW24
      L       MW44
      < >I
      =       M1.5
      O       M1.5
      O       M1.4
      O       M1.3
      A       I0.3
      JC      S6
    BEU
    C5:A      I0.4
      JC S1              //C5-1
      BEU
C6:A I0.4
      AN      T1
      JC      S1         //C6-1
      BEU
```

Network4:存储新状态,并跳转到新状态标签 Bn(为指行动作使用)

```
S1:   L       1
      JU      CS
S2:   L       2
      JU      CS
S3:   L       3
      JU      CS
S4:   L       4
      JU      CS
S5:   L       5
      JU      CS
S6:   L       6
      JU      CS
      BEU
CS:   T       #ZNr       //将当前状态数存入数据块中
      JL      Fauz
      JU      B0
```

```
        JU      B1
        JU      B2
        JU      B3
        JU      B4
        JU      B5
        JU      B6
Fauz:   BEU
Network5:动作
B0:     BEU
B1:     SET
        R       Q8.0        //关闭保险箱
        R       M   10.0    //关闭定时器
        L       0           //清除存储器
        T       MW   0
        BEU
B2:     L       MW200       //取数 1
        T       MW40
        BEU
B3:     L       MW200       //取数 2
        T       MW42
        BEU
B4:     L       MW200       //取数 3
        T       MW44
        BEU
B5:     SET
        S       Q8.0        //打开保险箱
        BEU
B6:     SET
        S       M10.0       //起动定时器
        BEU
```

3）经验法和状态图法的比较。经验法从初始口语描述出发，编程具有一定的盲目性。由于编程思路难于交流，所以在合作上具有一定的难度。由于口语描述的不准确性，所以增加了编程和调试的困难。

而状态图法是独立于技术执行过程和控制系统的建模行为。对于没有任何逻辑控制过程经验的人来说具有直观、易懂的特点。可以满足工程的需要，同时便于合作者之间的交流。总结其特点如下：

① 比口语描述更准确。

② 利用图形法更容易理解。

③ 独立于任何技术实现。

④ 易于项目组成员的交流和彼此的理解。

⑤ 易于文档整理。

本例如果用顺序功能图实现，由于程序的多分支，所以顺序功能图的绘制和实现程序比较复杂。顺序功能图和状态图两种方法各有其适合的范围，如果放弃恰当的方法，转而用不恰当的方法去分析问题，无异于舍近求远。

总之，状态图可以形成一个解决方式，能够直接转化成 PLC 程序，因此允许在编程时做阶段性的休整。执行这些模型不需要创造力和艺术表现，只不过是从一种形式到另一种形式的规则转换而已（可以被 PLC 执行)，顺序功能图的分析与程序实现也是如此。在编程时的一些模式决定了编程的重复性和严格性。曾有人询问一个工程经验丰富的德国教授，工程设计有何经验？他说，“没有什么经验，只要严格地按照编程定式去做就可以了”。就这么简单，当用户了解了顺序功能图和状态图的编程思路时，一定会同意上述的看法。

## 习题 Ⅱ

1. 如图 6-94 所示为大小球分类选择传送装置示意图。其工作原理如下：初始位置为机械臂在上部并在左边，机械臂下降（当磁铁压着大球时，限位开关 SQ2 断开，当压着的是小球时 SQ2 接通）时，如果是大球，则 Y1 吸住球并上升，至 SQ3 处右行，至 SQ5 后，下降至 SQ2，释放，之后上升至 SQ3，左移至初始位置。如果是小球则右行至 SQ4 后，下降、释放、上升、左移至初始位置。试根据工艺要求用顺序功能图和状态图两种方法进行系统设计。用梯形图实现顺序流程图程序，用 STL 语句实现状态图程序。

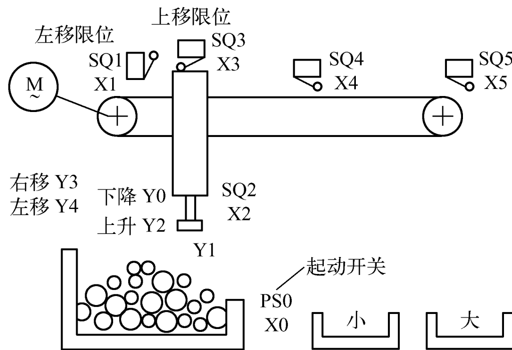


图 6-94　习题 1 图

2. 人行道按钮控制系统工作原理为：人行横道设有红、绿两盏信号灯（Q4. 0、Q4. 1），一般是红灯亮。路边设有按钮 SB1（I0. 7），行人横穿街道时需要按一下按钮（按钮起作用是在公路信号灯为绿灯时），4s 后公路黄灯亮（Q5. 1），再过 4s，公路红灯亮（Q5. 0），而此时人行道绿灯亮，使行人通过。10s 后，人行道红灯开始亮，再过 4s 公路黄灯亮。再过 4s 公路绿灯重新亮。试用状态图分析系统，并用语句表实现之。

# 第 7 章　结构化编程

## 7.1　概述

### 7.1.1　程序设计方法

程序设计中程序结构设计和数据结构设计是极为关键的内容，如何有效地利用 PLC 的内存资源和合理规划程序结构，将对编程周期、程序调试以及编程质量起到非常大的影响。在 STEP 7 中有三种程序设计方法：线性化编程、模块化编程和结构化编程。

1. 线性化编程

线性化编程是将整个用户程序放在循环控制组织块 OB1 中，在 CPU 循环扫描时执行 OB1 中的全部指令。其特点是结构简单、概念简单，但由于所有指令都在一个块中，程序的某些部分可能并不需要多次执行，而循环扫描则重复扫描所有指令，会造成资源浪费，效率低下。另一方面，某些相同或相近的操作需要多次执行，这样在编程中需要重复编写，同样会造成不必要的编程工作。再者，由于程序结构不清晰，会造成管理和调试的不方便。所以在编写大型程序时，要避免线性化编程。

2. 模块化编程

模块化编程是将程序根据功能分为不同的逻辑块，且每一逻辑块完成的功能不同。在 OB1 中可以根据条件调用不同的功能或功能块，其特点是易于分工合作，调试方便。由于逻辑块是有条件的调用，所以可以提高 CPU 的利用率。例如在泵站自动化监控工程中，根据泵站设备操作的具体过程，把 PLC 应用程序分成如下七个主要模块：

① 站备用投入模块（FC1），其功能是做好机组开机前泵站总体的准备工作，包括合主变 1 # 高压侧开关、合主变 1 # 低压侧开关、合主变 2 # 高压侧开关、合主变 2 # 低压侧开关等准备工作。

② 主机备用投入（FC20），其功能是做好机组开机前水泵自身的准备工作，即励磁通信和励磁状态检测、合冷却水电磁阀、油压检测、开防洪门等。

③ 主机开机（FC2），其功能是开加油阀，包括主电动机开关合（水泵起动）、检测定子电流等，若该电流过高，则报警退出。另外，还包括检测功率因数并根据实际情况进行调整。

④ 主机运行程序（FC16），该部分主要用于检测主机运行状态，不正常则报警。

⑤ 主机停机程序（FC3），其中包括主机断路器断，冷却水电磁阀断等。

⑥ 主机备用退出（FC35），其功能是关防洪门和关断加油阀。

⑦ 站备用退出（FC2），其功能是主变 1 # 低压侧开关断、主变 1 # 高压侧开关断等。

3. 结构化编程

结构化编程是将过程要求中类似或相关的任务归类，在功能或功能块中编程，形成通用

解决方案。通过不同的参数调用相同的功能或通过不同的背景数据块调用相同的功能块。例如：

- 传送带系统中所有交流电动机的通用逻辑控制块。
- 装配线机械中所有电磁线圈的通用逻辑控制块。
- 造纸机器中所有驱动装置的通用逻辑控制块。

结构化编程和模块化编程不同，通用的数据和代码可以共享。其特点是：编写通用模块，对不同的设备代入不同的地址和参数，使更多的设备或过程可以使用此通用模块，因此具有很高的编程和程序调试效率。程序结构层次清晰，标准化程度高。适合于复杂的控制任务，同时还支持多个程序员的协同工作。因为结构化编程必须对系统功能进行合理分析、分解和综合，所以对设计人员的要求较高。另外，当使用结构化编程方法时，需要对数据进行管理。

### 7.1.2　块的含义及调用

为支持结构化编程，在操作系统中包含了用户程序和系统程序，其中用户程序通常包括组织块（OB）、功能块（FB）、功能（FC）及数据块（DB）。系统块包括系统功能（SFC）、系统功能块（SFB）和系统数据块（SDB）。在块的调用中，调用者可以是各种的逻辑块，而被调用者是除 OB 块之外的逻辑块。调用功能块时，需为其指定背景数据块。

块的概念前面已经做过介绍。这里着重强调功能、功能块和数据块的含义。

**1. 功能（FC）**

功能是用户所编写的无固定存储区的块。它为不带“记忆”的逻辑块。所谓不带“记忆”表示没有背景数据块。当完成操作后，数据不能保持。这些数据为临时变量，对于那些需要保存的数据用户只能通过共享数据块（Share Block）来存储。

调用功能时，需用实参来代替形参。

**2. 功能块（FB）**

功能块是用户所编写的有固定存储区的块。FB 为带“记忆”的逻辑块。它有一个数据结构与功能块参数表完全相同的数据块（DB）。我们称该数据块为背景数据块（Instance Data Block）。当功能块被执行时，数据块被调用。功能块结束，调用随之结束。存放在背景数据块中的数据在 FB 块结束以后，仍能继续保持，具有“记忆”功能。

一个功能块可以有多个背景数据块，使功能块可以被不同的对象使用。

如前所述数据块包括共享数据块和背景数据块两种类型。共享数据块存储的是全局变量，所有的逻辑块都可以从共享数据块中读取数据；背景数据块则从属于某个功能块，用于传递参数。

块的调用即为子程序的调用，块可以嵌套调用，嵌套的层数与 CPU 的型号有关。

### 7.1.3　块的结构

我们知道块是由变量声明表和程序组成的。下面着重介绍变量声明表的含义。

**1. 变量声明表**

每个逻辑块都有变量声明表，而变量声明表用来说明块的局部数据。局部数据包括参数和局部变量两大类型。在不同的逻辑块中可以重复声明和使用同一个局部数据，因为它们在

每一个块中仅有效一次。

局部变量有两种：静态变量和临时变量。

参数则是在调用块和被调用块之间传递的数据，包括输入、输出和输入/输出变量。如表 7-1 所示为局部数据声明类型。

在组织块 OB 中，其调用是由操作系统来完成的，用户不能参与，所以 OB 块只有 L 堆栈（第 4 章中所提及的本地变量）的临时变量 TEMP。

在功能 FC 中，由于操作系统仅在 L 堆栈中给 FC 的临时变量分配存储区，块调用结束，变量消失，所以 FC 不能使用静态变量。

**表 7-1　局部数据声明类型**

| 变　量　名 | 类　　型 | 说　　明 |
|---|---|---|
| 输入 | IN | 为调用逻辑块提供数据，输入给逻辑块 |
| 输出 | OUT | 从逻辑块中输出数据结果 |
| 输入/输出 | IN_OUT | 参数值可以输入，也可以输出 |
| 静态变量 | STAT | 静态变量存储于背景数据块中，块调用结束后，变量被保留 |
| 临时变量 | TEMP | 临时变量存储于 L 堆栈中，块执行结束后，变量消失 |

在功能块 FB 中，操作系统为参数和静态变量分配的存储空间是背景数据块。

功能和功能块的调用必须用实参代替形参，这是为什么呢？因为形参是在功能或功能块的变量声明表中定义的。为保证功能或功能块对同一类设备的通用性，在编程中不能使用实际对应的存储区地址参数，而是使用抽象参数，这就是形参。形象地说功能和功能块是将同一类设备封装了起来。而块在调用时，必须将实际参数（实参）替代形参，从而可以通过功能或功能块实现对具体设备的控制。即所谓的从“抽象”到“具体”。这里必须注意：实参的数据类型必须与形参一致。

**2. 局部数据的数据类型**

在变量声明表中，必须明确局部数据的数据类型，这样才能分配相应的存储空间。局部数据可以是基本数据类型，也可以是复合数据类型，还可以是参数类型。这些数据类型前面已经有所介绍。

## 7.2　功能和功能块编程及调用举例

功能和功能块的编程步骤如下：

1）定义局部变量。首先定义形参和临时变量名，功能块还须定义静态变量。之后确定变量的类型及变量注释。

2）编写执行程序，在编程中若使用变量名，则变量名标识为前缀“#”加变量名。若使用全局符号则显示为全局符号加引号的形式。

对于 S7－300，操作系统分配给每一个 OB 的局部数据区的最大数量为 256B。OB 的调用自己占去 20B 或 22B，则还剩下最多 234B 可分配给 FC 或 FB。如果块中定义的局部数据的数量大于 256B，该块将不能下装到 CPU 中。在下装过程中将出现错误提示：“The block could not be copied”。如果单击错误信息框中的“Details”按钮，将弹出帮助信息：“Incor-

rect local data length”。利用“Reference Data”工具可查看程序所占用的局部数据区的字节数（包括总的字节数和每次调用所占用的字节数）。

### 7.2.1 功能编程及举例

**例 1** 设计故障指示系统：故障信号 I1.3（Disturb_ input）出现时，在操作面板上用一个指示灯 Q4.3（LED）来指示。指示灯以 2Hz 的频率闪烁。系统用应答输入 I 1.2（Acknowledge）来检测故障的存在，如果故障已排除，则指示灯停止闪烁；若故障仍然存在，则指示灯转换为常亮状态直到故障被排除。故障指示波形如图 7-1 所示。

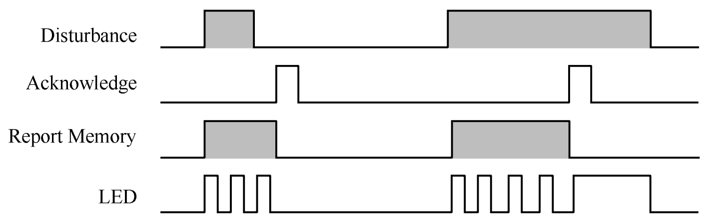



图 7-1 故障指示波形

故障指示系统的功能块图程序如图 7-2 所示。

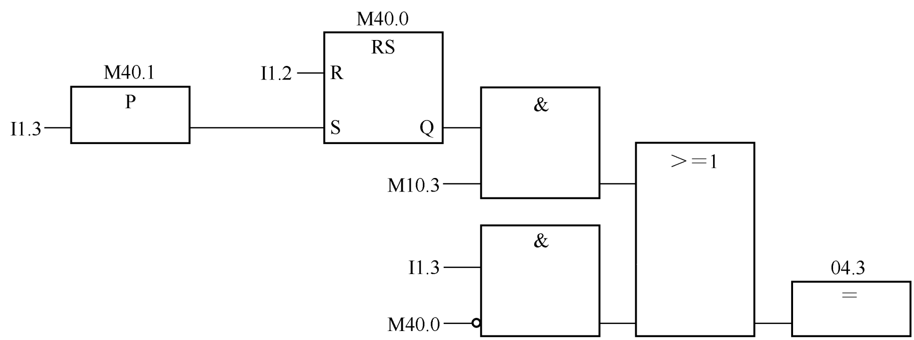



图 7-2 故障指示系统的功能块图程序

说明：故障信号的出现采用置位指令将其锁存在故障标志位 M40.0 中。在置位指令之前，对故障信号做 RLO 边沿检测处理，这样在故障应答信号输入后可立即复位故障标志位。如果故障标志被置位（且该信息未被应答），则利用功能块图上面的与逻辑指令使指示灯闪烁。与逻辑指令的一个输入端为时钟信号标志位 M10.3，另一输入端为故障标志位，这样就可以用来决定时钟信号是否输出。功能块图下面的与逻辑指令的作用是：当应答过后故障仍然存在时，令 LED 常亮。

现在将上述系统用功能 FC 封装。如图 7-3 所示为局部数据的定义。其中输入变量 Disturb_input（故障输入）取代 I1.3，Acknowledge 取代 I1.2，Flash_freq 取代 M10.3；输出变量 Display 取代 Q4.3；输入/输出变量 Edge_mem_bit 取代 M40.1，Report_memory 取代 M40.0，因为对故障信息标志位既要读（扫描）又要写（set/reset），所以它被定义为 IN_OUT 型参数。这样带局部数据的 FC1 程序如图 7-4 所示。

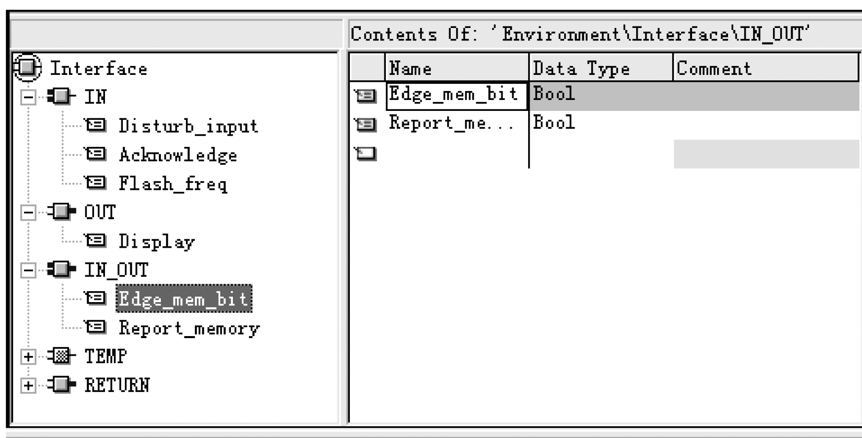



图 7-3　局部数据的定义

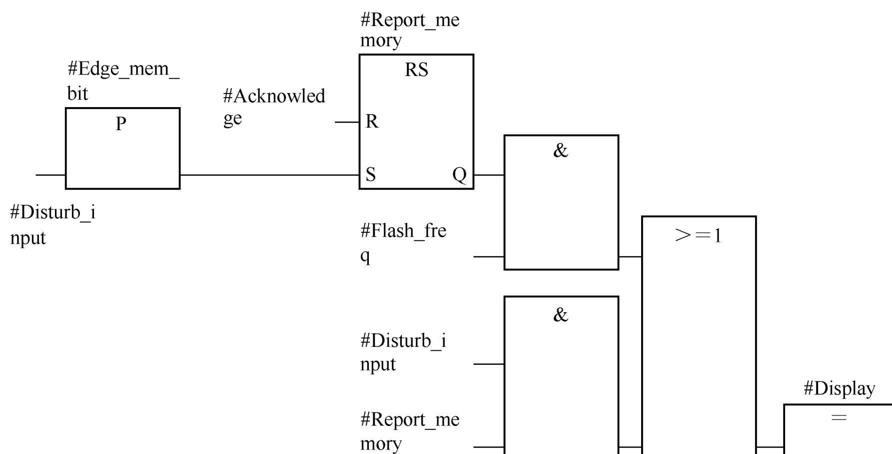



图 7-4　带局部数据的 FC1 程序

语句表的调用如下：

```
CALL    FC1
Disturb_input     : = I1. 3
Acknowledge       : = I1. 2
Flash_freq        : = M10. 3
Display           : = Q4. 3
Edge_mem_bit      : = M40. 1
Report_memory     : = M40. 0
```

程序封装在 FC1 中。OB1 中用实参代替形参的 FC1 的封装形式和两次调用如图 7-5 所示。这里，故障信号分别为 I0. 3 和 I0. 4，故障显示分别为 Q4. 3 和 Q4. 4，中间存储位各不相同。这样，使用不同的参数调用 FC1，达到监控故障的相近功能。

**例 2**　设计模拟量读入程序 FC10。其中模拟量输入模块的起始地址、通道数、存储数据块号及数据在数据块中的存储起始位置均可变。

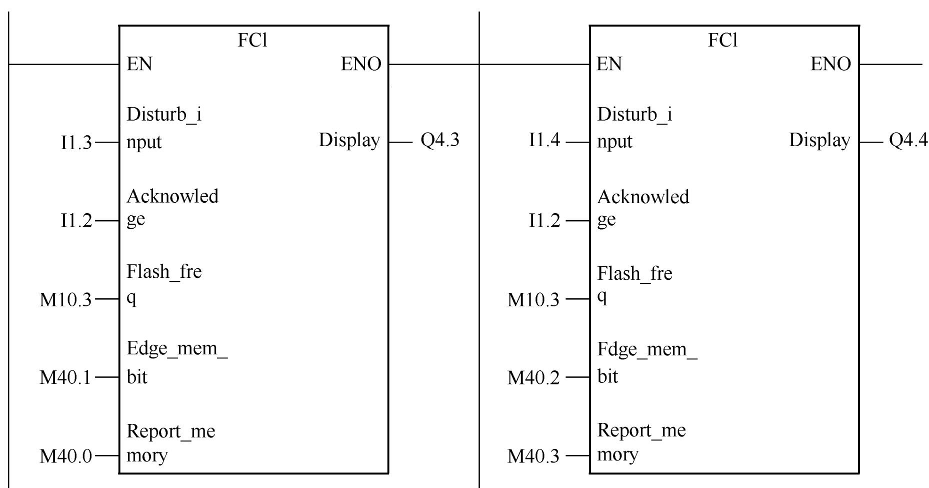



图 7-5　将实参代替形参的 FC1 的封装形式和两次调用

FC10 变量声明表如图 7-6 所示，定义了 4 个输入变量：模拟量输入模块的起始地址（PIW_Addr）、通道数（CH_Len）、存储数据块号（DB_No）及数据块中的存储起始位置（DBW_Addr），其类型均为 Int。

Interface
- IN
- OUT
- IN_OUT
- TEMP
- RETURN

Contents Of: 'Environment\Interface\IN'

| Name | Data Type | Comment |
| --- | --- | --- |
| PIW_Addr | Int | |
| CH_Len | Int | |
| DB_No | Int | |
| DBW_Addr | Int | |

图 7-6　FC10 变量声明表

语句表程序如下：

```
      L     #DB_No
      T     LW0
      OPEN DB[LW0]          //打开存储数据块
      L     #PIW_Addr
      SLD   3               //形成模拟量输入模块地址指针
      T     LD4             //在临时本地数据中存储地址指针
      L     #DBW_Addr
      SLD   3               //形成数据块存储地址指针
      T     LD8             //在 LD8 中存储地址指针
      L     #CH_LEN         //在 LW0 中装载读入通道数作为循环计数器
NEXT：T     LW0
      L     LD4
      LAR1                  //将模拟量模块地址指针装入地址寄存器
```

```
L       PIW[AR1,P#0.0]
T       LW2                 //将累加器中模拟量模块内容暂存入 LW2 中
L       LD8
LAR1                        //将数据块存储地址指针装入地址寄存器
L       LW2
T       DBW[AR1,P#0.0]      //将模拟量内容存入数据块中
L       LD4
+       L#16
T       LD4                 //调整模拟量模块地址指针
L       LD8
+       L#16
T       LD8                 //调整数据块存储地址指针
L       LW0
LOOP NEXT                   //循环计数器 LW0 减 1,如若不为零,继续循环,否则结束
```

程序中使用了存储器间接寻址方式。其中临时本地数据为 LW0、LD4、LD8 等，能否将其定义为临时变量 Temp 呢？答案是否定的，因为临时变量不能用于存储器间接寻址。

程序的调用如下：

```
CALL        FC10
PIW_Addr          :=256
CH_Len            :=8
DB_No             :=30
DBW_Addr          :=10
```

程序可以实现将地址为 256 的模拟量输入模块的 8 个信号读入到数据库 DB30.DBW10 开始的字单元中。

### 7.2.2 功能块编程及举例

FB 块带有一个存储区，即背景数据块。那么背景数据块如何生成呢？有以下两种方法：

① 在调用 FB 并为它指定一个背景数据块后，如果该数据块并不存在，则弹出以下提示信息："Instance data block DB x does not exist. Do you want to generate it?"。单击"Yes"按钮可自动生成一个新的背景数据块。

② 创建一个新的数据块时，选择其属性为"Data block referencing a function block"。

功能块 FB 的优点：当编写 FC 的程序时，用户必须寻找空的标志区或数据区来存储需保持的数据，并且要自己编写程序来保存它们。而 FB 的静态变量可由 STEP 7 的软件自动保存。将例 1 改编为 FB 功能块。要求保存变量"Edge_mem_bit"和"Report_memory"。

背景数据块可以保存静态变量，当块退出时，它们仍然保持。所以我们将变量"Edge_mem_bit"和"Report_memory"设置为静态变量，这样当用 FB1 块实现 FC1 的功能，并用静态变量"Edge_mem_bit"和"Report_memory"来代替原来的形式参数，将可以省略两个形式参数，简化了块的调用。FB 变量声明表如图 7-7 所示。

FB 中的程序和 FC 中的程序内容完全相同，如图 7-4 所示。FB 功能块的封装形式及调用如图 7-8 所示。

Contents Of: 'Environment\Interface\IN'

Interface
- IN
  - Disturb_input
  - Flash_freq
  - Acknowledge
- OUT
  - Display
- IN_OUT
- STAT
  - Edge_mem_bit
  - Report_memory
- TEMP

| Name | Data Type | Address | Initial | Exclusion ad |
|---|---|---|---|---|
| Disturb_input | Bool | 0.0 | FALSE | |
| Flash_freq | Bool | 0.1 | FALSE | |
| Acknowledge | Bool | 0.2 | FALSE | |
| | | | | |

图 7-7　FB 变量声明表

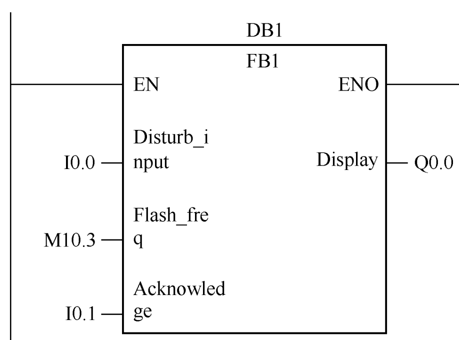



图 7-8　FB 功能块的封装形式及调用

生成的背景数据块 DB1 如图 7-9 所示。注意要对系统进行监控，必须下载数据块内容。

DB1 -- S7_Pro3\SIMATIC 300 Station\CPU312C(1)

| | Address | Declaration | Name | Type | Initial value | Actual valu |
|---|---|---|---|---|---|---|
| 1 | 0.0 | in | Disturb_input | BOOL | FALSE | FALSE |
| 2 | 0.1 | in | Flash_freq | BOOL | FALSE | FALSE |
| 3 | 0.2 | in | Acknowledge | BOOL | FALSE | FALSE |
| 4 | 2.0 | out | Display | BOOL | FALSE | FALSE |
| 5 | 4.0 | stat | Edge_mem_bit | BOOL | FALSE | FALSE |
| 6 | 4.1 | stat | Report_memory | BOOL | FALSE | FALSE |

图 7-9　背景数据块 DB1

## 7.3　FC 和 FB 程序设计实例

### 7.3.1　任务描述

工业搅拌过程如下：两种配料（A、B）在一个混合罐中由搅拌器混合在一起，之后通过排料阀排出。工业搅拌示意图如图 7-10 所示。系统分为 4 个区：配料 A、配料 B、搅拌

区、排料区。电动机和泵有 3 台：配料 A 进料泵、配料 B 进料泵、搅拌电动机。阀门有五个：配料 A 入口阀、配料 A 进料阀、配料 B 入口阀、配料 B 进料阀、排料阀。

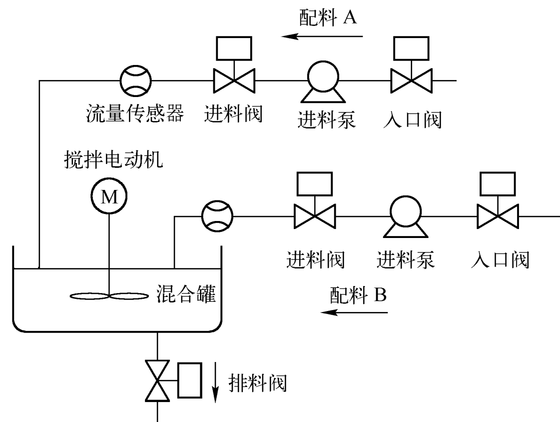



图 7-10　工业搅拌示意图

各个区域的功能如下：

配料 A 和配料 B 的每个配料管都配有一个入口阀和进料阀，还有一个进料泵。配料管中还有流量传感器，检测是否有配料流过。区域功能如下：

① 进料泵 ：当罐的液面传感器指示混合罐装满后，进料泵必须关闭。

② 进料泵 ：当排料阀打开时，进料泵同样也要关闭。

③ 阀门 ：在起动进料泵 1 s 后，必须打开入口阀和进料阀。

④ 阀门 ：在进料泵停止后，阀门必须关闭，防止配料泄露。

⑤ 故障检测 ：进料泵起动 7 s 之后，流量传感器会报溢出。

⑥ 故障检测 ：进料泵运行时，若流量传感器没有流量信号，则进料泵关闭。

⑦ 维护 ：进料泵起动次数大于 50 次，必须维护。

搅拌区的混合罐中装有 3 个传感器：罐装满传感器（装满之后，触点断开）、罐不空传感器、罐液体最低限位（达到最低限位，触点关闭）。搅拌区功能如下：

① 搅拌电动机 ：当液面指示“液面高度低于最低限位”时，或者排料阀打开时，搅拌电动机必须停止。

② 故障检测 ：如果搅拌电动机在起动后 10s 内没有达到电动机的额定转速，则电动机必须断开。

③ 维护 ：搅拌电动机的起动次数超过 50 次，进行维护。

排料区中成品的排出由螺线管阀门控制。排料区的功能如下：

① 罐空时，阀门必须关闭；

② 当搅拌电动机工作时，或者罐空时排料阀必须关闭，这一要求与搅拌区功能的第一项要求一致。

在进行系统设计之前，首先分析系统功能，可以看出系统有多台电动机和多个阀门，如果直接用线性化或模块化编程，会有较多的重复编程，而用结构化编程可以减少工作量。

将电动机设计封装在 FB1 中，以不同的数据块分别表示不同的电动机：配料 A 的进料泵（DB1）、配料 B 的进料泵（DB2）、搅拌电动机（DB3）。阀门用 FC 来封装，分别表示

配料 A 和 B 的入口阀、进料阀和排料阀。分层结构如图 7-11 所示。

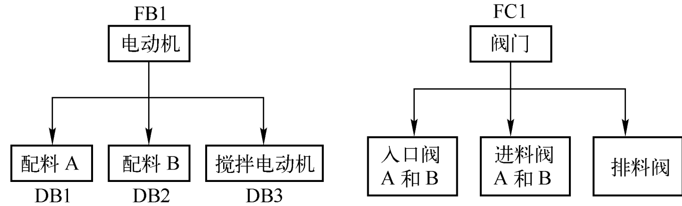



图 7-11　分层结构图

## 7.3.2　建立符号表

表 7-2 为系统符号表。

**表 7-2　系统符号表**

| 进料泵、搅拌电动机符号地址 | | | |
|---|---|---|---|
| 符号名 | 地址 | 数据类型 | 描述 |
| FeedA_start | I0.0 | BOOL | 起动配料 A 的进料泵 |
| FeedA_stop | I0.1 | BOOL | 停止配料 A 的进料泵 |
| FlowA | I0.2 | BOOL | 配料 A 流动 |
| InletVA | Q4.0 | BOOL | 起动配料 A 的入口阀 |
| PumpVA | Q4.1 | BOOL | 起动配料 A 的进料阀 |
| PumpAL_on | Q4.2 | BOOL | 配料 A 进料泵运行指示灯 |
| PumpAL_off | Q4.3 | BOOL | 配料 A 进料泵停止指示灯 |
| PumpA | Q4.4 | BOOL | 配料 A 进料泵运行 |
| FaultAL | Q4.5 | BOOL | 进料泵 A 故障指示灯 |
| MaintAL | Q4.6 | BOOL | 进料泵 A 维护指示灯 |
| FeedB_start | I0.3 | BOOL | 起动配料 B 的进料泵 |
| FeedB_stop | I0.4 | BOOL | 停止配料 B 的进料泵 |
| FlowB | I0.5 | BOOL | 配料 B 流动 |
| InletVB | Q5.0 | BOOL | 起动配料 B 的入口阀 |
| PumpVB | Q5.1 | BOOL | 起动配料 B 的进料阀 |
| PumpBL_on | Q5.2 | BOOL | 配料 B 进料泵运行指示灯 |
| PumpBL_off | Q5.3 | BOOL | 配料 B 进料泵停止指示灯 |
| PumpB | Q5.4 | BOOL | 配料 B 进料泵运行 |
| FaultBL | Q5.5 | BOOL | 进料泵 B 故障指示灯 |
| MaintBL | Q5.6 | BOOL | 进料泵 B 维护指示灯 |
| AgitatorR | I1.0 | BOOL | 搅拌电动机相应信号 |
| Agitator_start | I1.1 | BOOL | 搅拌电动机起动按钮 |
| Agitator_stop | I1.2 | BOOL | 搅拌电动机停止按钮 |
| Agitator | Q8.0 | BOOL | 搅拌电动机起动 |
| AgitatorL_on | Q8.1 | BOOL | 搅拌电动机运行指示灯 |
| AgitatorL_off | Q8.2 | BOOL | 搅拌电动机停止指示灯 |
| FaultGL | Q8.3 | BOOL | 搅拌电动机故障指示灯 |
| MaintGL | Q8.4 | BOOL | 搅拌电动机维护指示灯 |

（续）

| 排料阀的符号地址 | | | |
|---|---|---|---|
| 符号名 | 地址 | 数据类型 | 描述 |
| Drain_open | I0.6 | BOOL | 打开排料阀按钮 |
| Drain_closed | I0.7 | BOOL | 关闭排料阀按钮 |
| Drain | Q9.5 | BOOL | 起动排料阀 |
| DrainL_on | Q9.6 | BOOL | 排料阀运行指示灯 |
| DrainL_off | Q9.7 | BOOL | 排料阀停止指示灯 |
| 传感器及液面显示符号地址 | | | |
| 符号名 | 地址 | 数据类型 | 描述 |
| Tank_Lmax | I1.3 | BOOL | 混合罐未满传感器 |
| Tank_Amin | I1.4 | BOOL | 混合罐液面高于最低限位传感器 |
| Tank_Nemp | I1.5 | BOOL | 混合罐非空传感器 |
| 其他符号地址 | | | |
| 符号名 | 地址 | 数据类型 | 描述 |
| EM_stop | I1.6 | BOOL | 紧急停机开关 |
| ResetM | I1.7 | BOOL | 复位维护指示灯 |

### 7.3.3　生成电动机 FB

电动机的通用 FB 示意图如图 7-12 所示。

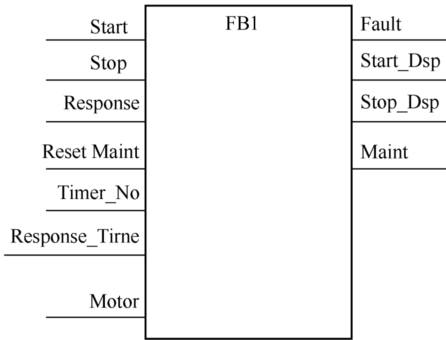



图 7-12　电动机的通用 FB 示意图

定义 FB 的变量声明表如图 7-13 所示。其中输入量为 Start、Stop、Response、Reset_Maint、Timer_No 和 Response_Timer；输出量为 Fault、Start_Dsp、Stop_Dsp 和 Maint；输入/输出量为 Motor；静态变量为 Start_Edge 和 Starts。

FB 实现的功能如下：

① 起动和停止电动机，起动和停止时点亮相应指示灯。

② 设备起动，则监控定时器起动，定时时间到且未接收到设备相应信号则将电动机停止，并且指示故障。

③ 起动次数大于 50 次，则维护指示灯亮。

④ 设备的起、停有互锁条件，在 OB1 中体现。

电动机 FB 的梯形图程序如图 7-14 所示。

Contents Of: 'Environment\Interface\IN'

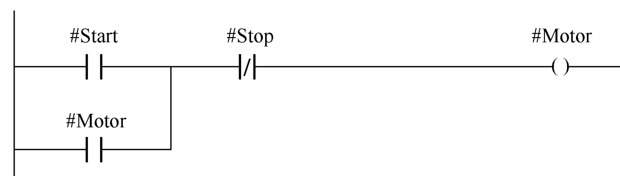
- Interface
  - IN
    - Start
    - Stop
    - Response
    - Reset_Maint
    - Timer_No
    - Response_Time
  - OUT
    - Fault
    - Start_Dsp
    - Stop_Dsp
    - Maint
  - IN_OUT
    - Motor
  - STAT
    - Start_Edge
    - Starts
  - TEMP

| Name | Data Type | Address |
|---|---|---|
| Start | Bool | 0.0 |
| Stop | Bool | 0.1 |
| Response | Bool | 0.2 |
| Reset_Maint | Bool | 0.3 |
| Timer_No | Timer | 2.0 |
| Response_Time | S5Time | 4.0 |

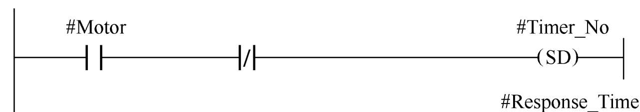
图 7-13　FB 的变量声明表

FB1: 电动机

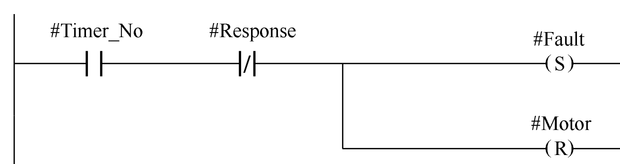
Network 1: 起动和停止电动机



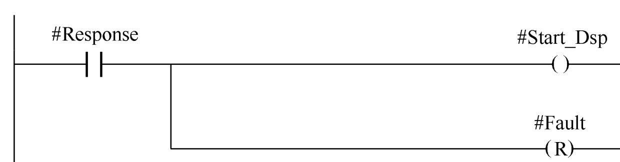

Network 2: 起动监控定时器




Network 3: 监控时间到且没有反应则故障指示灯亮，停止电动机




Network 4: 点亮指示灯并复位故障




Network 5: 断开指示灯

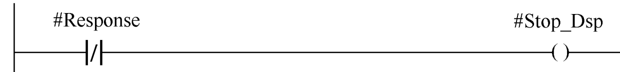



图 7-14　电动机 FB 的梯形图程序

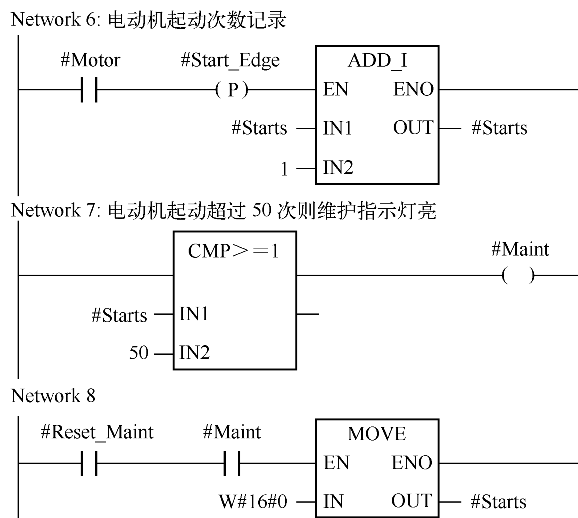



图 7-14　电动机 FB 的梯形图程序（续）

说明：程序的 Network1 为起动和停止电动机；Network2 到 Network5 为故障处理程序；Network6 到 Network8 为电动机维护。

### 7.3.4　生成阀门 FC

阀门的功能如下：

① 打开和关闭阀门。

② 开启和关闭阀门时，相应指示灯亮。

③ 互锁状态在 OB1 中体现。

FC 变量声明表如图 7-15 所示。阀门的通用 FC 示意图如图 7-16 所示。其中输入变量为 Open、Close 和 Value；输出变量为 Open_Dsp 和 Close_Dsp。

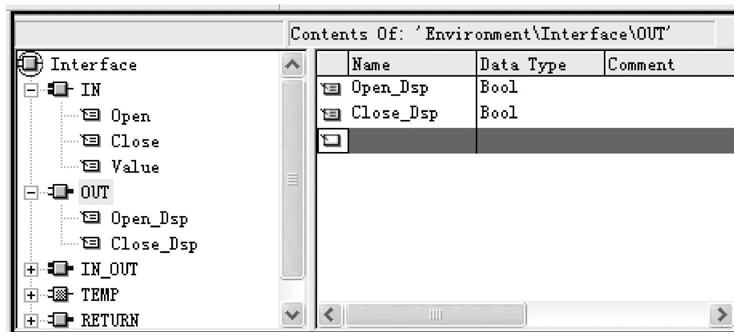



图 7-15　FC 变量声明表

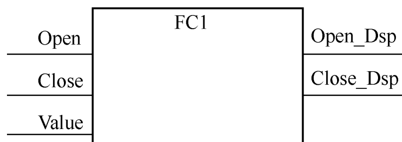



图 7-16　阀门的通用 FC 示意图

阀门 FC 的梯形图程序如图 7-17 所示。

说明：程序 Network1 为打开和关闭阀门；Network2 到 Network3 为相关指示灯显示。

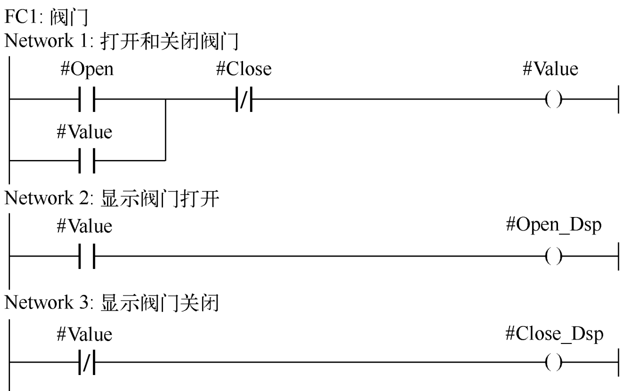



图 7-17　阀门 FC 的梯形图程序

### 7.3.5　生成 OB1

OB1 实现功能如下：

① 完成互锁功能：用#Enable_Mortor 来控制电动机或泵的起动，用#Enable_Value 来控制阀门的起动。

② 提供监控定时时间。

③ 为 FB 提供不同的数据块。

OB1 组织块的临时（局部）变量定义如图 7-18 所示。

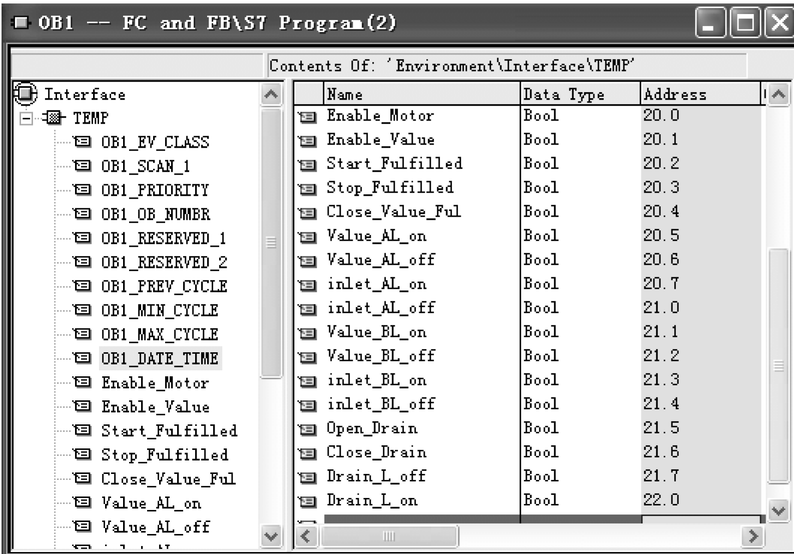
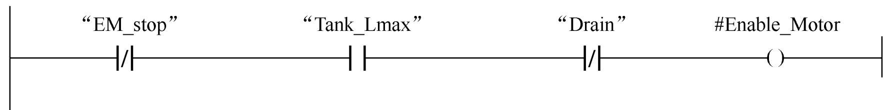



图 7-18　OB1 组织块的临时变量

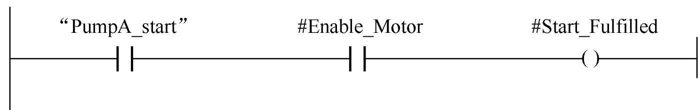
图中有使能电动机变量（Enable_Motor）、使能阀门变量（Enable_Value）、起动电动机信号使能变量（Start_Fulfilled）、停止电动机信号使能变量（Stop_Fulfilled）、关闭阀门信号使能变量（Close_Value_Ful）和指示信号等。OB1 梯形图程序如图 7-19 所示。

OB1：主程序

Network 1：进料泵的互锁：在混合罐未满且排料阀未开前提下



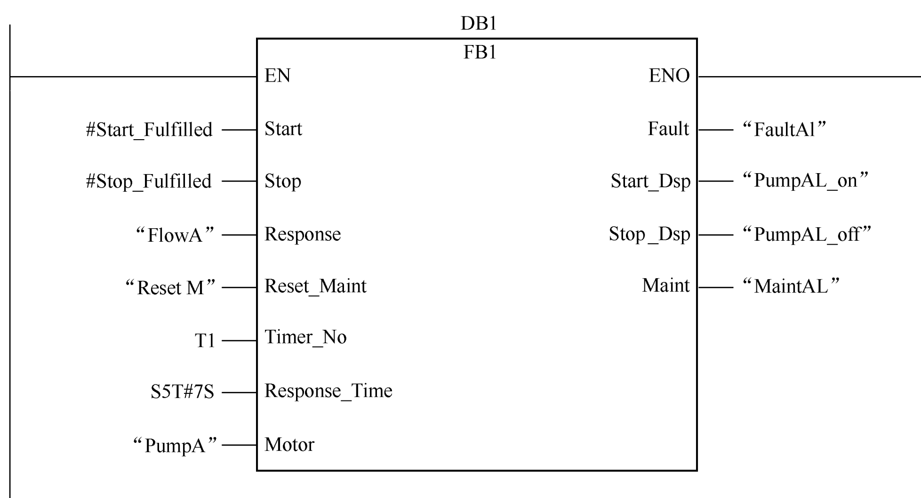

Network 2：使能进料泵 A 的起动



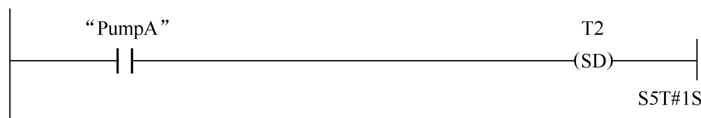

Network 3：使能进料泵 A 的停止




Network 4：配料 A 进料泵，调用 DB1




Network 5：进料泵打开 1s 之后，使能阀门打开

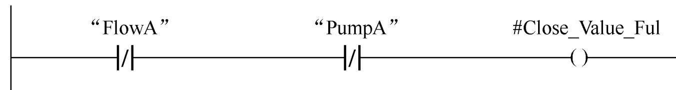



Network 6

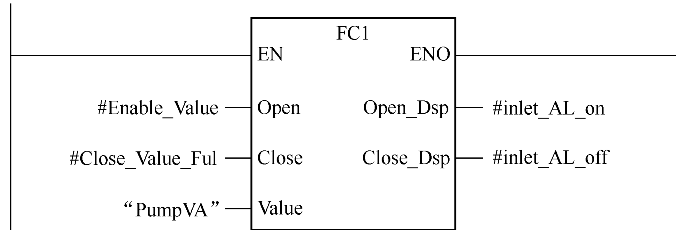



图 7-19　OB1 梯形图程序

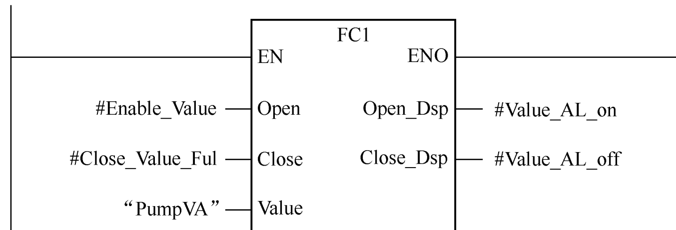
Network 7: 使能阀门关闭（停止进料后）



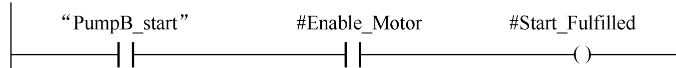

Network 8: 配料 A 入口阀控制



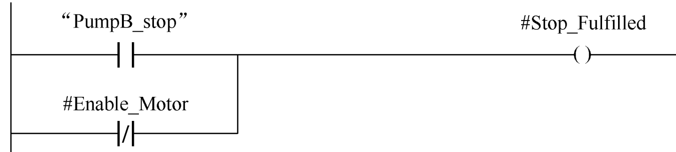

Network 9: 配料 A 进料阀控制




Network 10: 使能进料泵 B 的起动




Network 11: 使能进料泵 B 的停止




Network 12: 配料 B 进料泵，调用 DB2

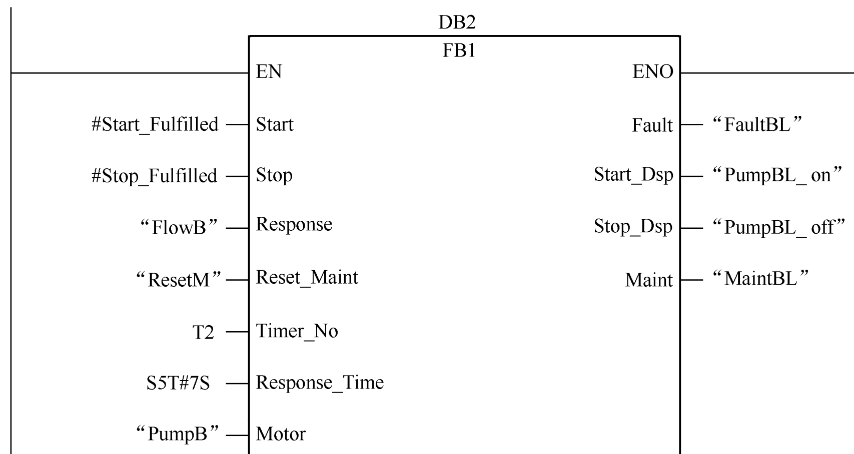



图 7-19　OB1 梯形图程序（续）

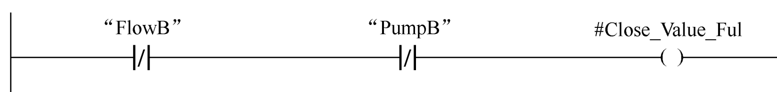
Network 13: 进料泵 B 打开 1s 之后，使能阀门打开



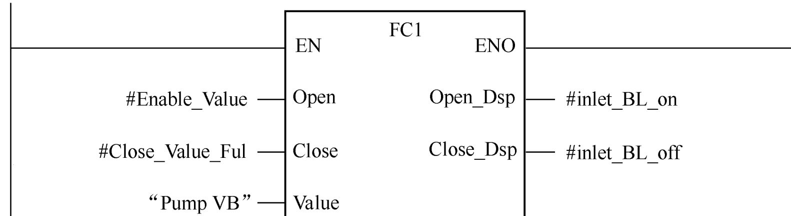

Network 14



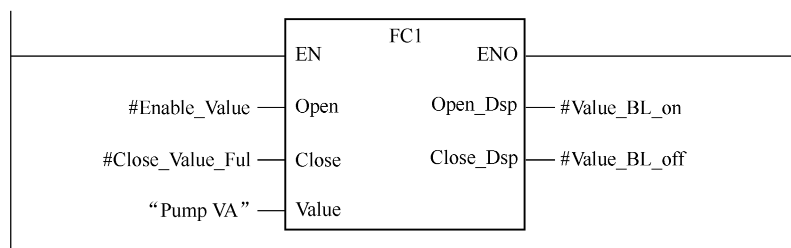

Network 15: 使能阀门关闭（停止进料后）



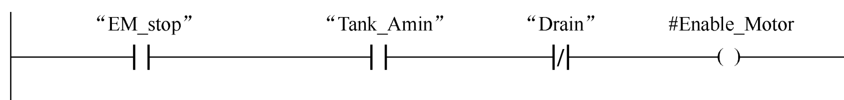

Network 16: 配料 B 入口阀控制



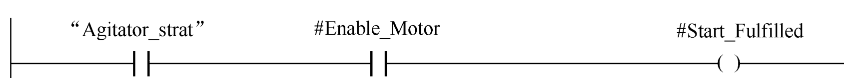

Network 17: 配料 B 进料阀控制




Network 18: 搅拌器互锁：在混合罐液面高于最低限且排料阀未开前提下




Network 19: 使能搅拌器的起动




Network 20: 使能搅拌器的停止

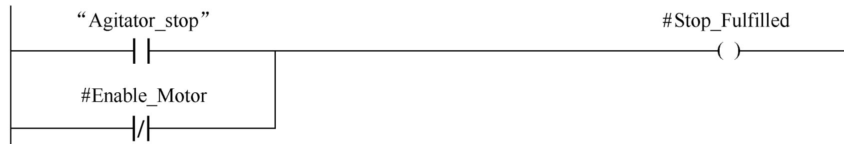



图 7-19　OB1 梯形图程序（续）

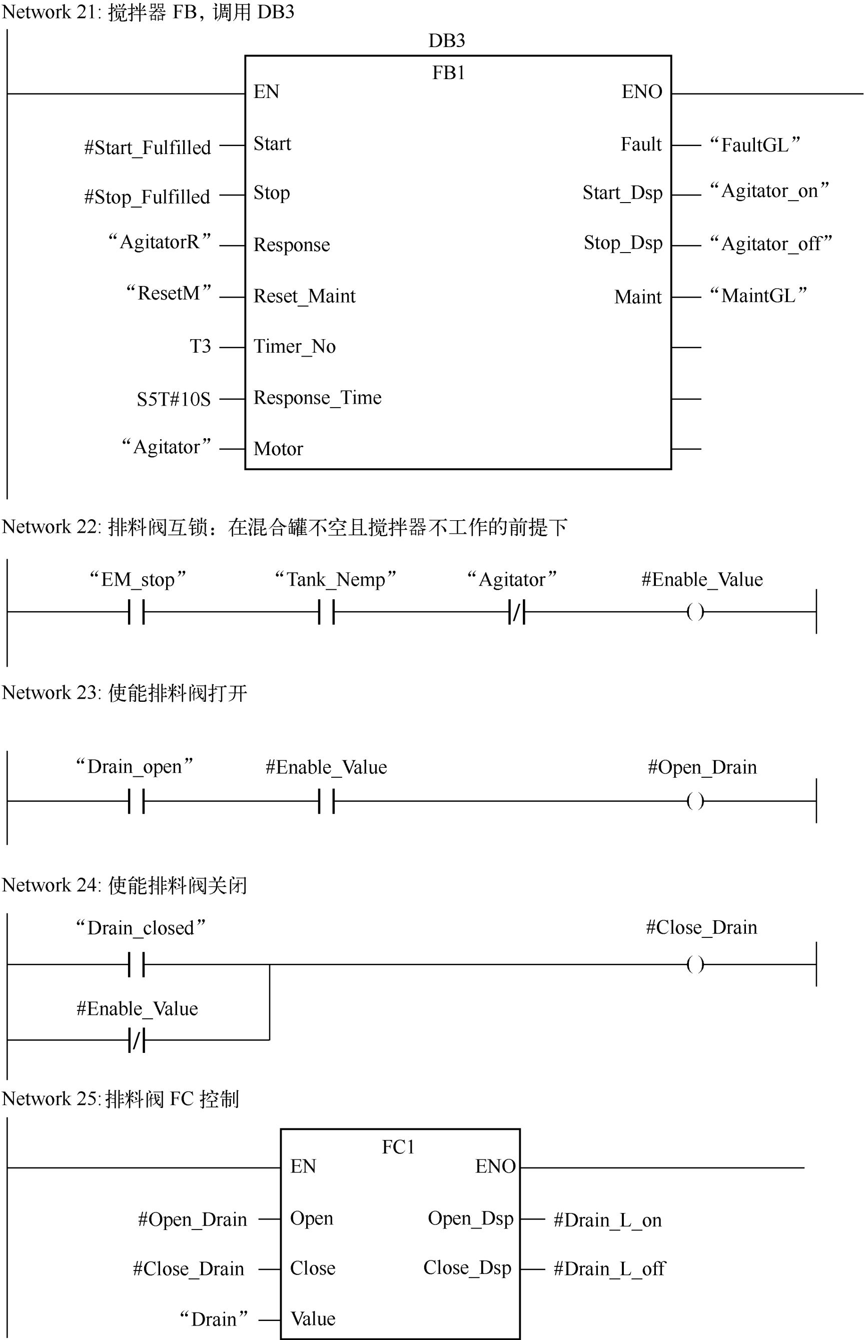



图 7-19　OB1 梯形图程序（续）

说明：程序的 Network1 到 Network9 为配料 A 的进料泵、入口阀及进料阀的控制；Network10 到 Network17 为配料 B 的进料泵、入口阀及进料阀的控制；Network18 到 Network25 为搅拌器及排料阀的控制。

## 习题

1. 试用 FC 封装风扇控制。要求：电动机起动（Engine_On）后，起动延时定时器（Timmer_No）延时（Time），之后将风扇起动（Fan_On）。

2. 试用 FB 封装发动机的控制，发动机有两种类型：汽油机和柴油机，要求：输入为起动（Switch_On）、停止（Switch_Off）、故障（Failure）、复位故障（Reset）和转速设置（Actual_Speed）；静态变量为转速预设值（Preset_Speed）；输出为运行（Engine_On）、转速达到设置转速指示灯（Achieve_Speed_L）和故障指示灯（Failure_L）。编程要求起动和停止按钮使运行线圈工作或停止。转速达到给定转速时，显示转速达到预设值的指示灯亮。故障使故障指示灯亮。复位故障后，故障指示灯灭。

3. 将走马灯封装在 FC 中，输入为起动和停止按钮（Start、Stop）、左转和右转按钮（Left、Right）、定时间隔（Time）、初值设置（initial）以及移位间隔（interval）；输出为 8 位（show）。

# 第 8 章　组织块及系统功能的使用

有人说 STEP 7 难学，并非指编程语言难学，而是指 S7－300/400 中有许多功能强大的组织块和系统功能难以掌握。本章将主要介绍其用法。组织块及系统功能相当于 S7 的子程序，掌握并能使用好这些子程序，无疑像站在巨人的肩膀上。

## 8.1　组织块

组织块（Organization Block）又称为 OB，STEP 7 提供了大量的组织块用于执行用户程序。OB 是 CPU 操作系统与用户程序间的接口。OB 被嵌在用户程序中，根据某个事件的发生，执行相应的中断，自动调用相应的 OB。例如循环中断 OB10、硬件错误中断 OB40、机架故障 OB86 等。

当操作系统调用其他组织块时，循环程序 OB1 的执行被中断，因为 OB1 的优先级最低，所以任何其他的 OB 都可以中断主程序并执行自己的程序，执行完毕后从断点处开始恢复执行 OB1。当有比当前执行的程序优先级更高的 OB 被调用时，程序将中止当前正在运行的 OB，转而调用更高优先级的 OB。操作系统为被中断的块保存全部的寄存器堆栈。当返回被中断的块时，寄存器的信息被恢复。此为压栈、出栈过程。

OB 具有不同的优先级，优先级的范围从 1～29，其中“1”优先级最低，“29”优先级最高。每一个 OB 在执行程序的过程中可以被更高优先级的事件中断。具有同等优先级的 OB 不能相互中断，而是按照发生的先后顺序执行。OB 的类型与默认优先级如表 8-1 所示。

表 8-1　OB 的类型与默认优先级

| OB 编号 | 起动事件 | 默认的优先级 |
|---|---|---|
| OB1 | 主程序，循环执行 | 1 |
| OB10～OB17 | 8 个日期时间中断 | 2 |
| OB20～OB23 | 4 个延时中断 | 3～6 |
| OB30～OB38 | 9 个循环中断 | 7～15 |
| OB40～OB47 | 8 个硬件中断 | 16～23 |
| OB55 | 状态中断（DPV1 中断） | 2 |
| OB56 | 刷新中断 | 2 |
| OB57 | 制造厂商用特殊中断 | 2 |
| OB60 | 调用 SFC 35 时起动，多处理器中断 | 25 |
| OB61～OB64 | 4 个周期同步中断 | 25 |
| OB70 | I/O 冗余故障（只对于 H CPU） | 25 |
| OB72 | CPU 冗余故障（只对于 H CPU） | 28 |
| OB73 | 通信冗余故障（只对于 H CPU） | 25 |

（续）

| OB 编号 | 起动事件 | 默认的优先级 |
| --- | --- | --- |
| OB80 | 时间错误 | 26，起动时为 28 |
| OB 编号 | 起动事件 | 默认的优先级 |
| OB81 | 电源故障 | 26，起动时为 28 |
| OB82 | 诊断中断 | 26，起动时为 28 |
| OB83 | 模板插/拔中断 | 26，起动时为 28 |
| OB84 | CPU 硬件故障 | 26，起动时为 28 |
| OB85 | 程序故障 | 26，起动时为 28 |
| OB86 | 扩展机架、DP 主站系统或分布式 I/O 从站故障 | 26，起动时为 28 |
| OB88 | 过程中断 | 28 |
| OB90 | 暖或冷起动、删除块或背景循环 | 29 |
| OB100 | 暖起动 | 27 |
| OB101 | 热起动 | 27 |
| OB102 | 冷起动 | 27 |
| OB121 | 编程错误 | 与被中断的块在同一优先级 |
| OB122 | I/O 访问故障 | 与被中断的块在同一优先级 |

## 8.2　循环处理的主程序 OB1

循环处理的主程序 OB1 的优先级最低，即除 OB90（背景组织块）以外，其他所有 OB 均可中断 OB1 的执行。CPU 起动完毕后，操作系统就调用 OB1，并且循环执行 OB1 的程序，读取当前输入模块的信号状态，刷新输入映像区并接收全局数据。每一次 OB1 程序执行完后，操作系统发送全局数据，传送输出映像区数据到输出模块。

在 STEP 7 中，可以设置每次处理 OB1 的最长时间和最短时间。具体方法是：在硬件组态中，点击“CPU Properties”，弹出 CPU 属性窗口并选择“Cycle/Clock Memory”选项页，如图 8-1 所示。设置最小循环时间，则 CPU 循环系统将延时达到此时间后才开始下一次 OB1 的执行。

另外，可以在“Cycle/Clock Memory”选项页中设置“Clock Memory”，选中如图 8-1 所示选择框就可激活该功能，并且在“Memory Byte”中输入存储字节 MB 的地址，如 MB10（输入 10 即可），此时 MB10 各位的作用是产生不同频率的方波信号。如果用户在硬件配置里选择了该项功能，就可以在程序里调用这些特殊的位。Clock Memory 各位的周期及频率如表 8-2 所示。

**表 8-2　Clock Memory 各位的周期及频率**

| 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 周期/s | 2 | 1.6 | 1 | 0.8 | 0.5 | 0.4 | 0.2 | 0.1 |
| 频率/Hz | 0.5 | 0.625 | 1 | 1.25 | 2 | 2.5 | 5 | 10 |

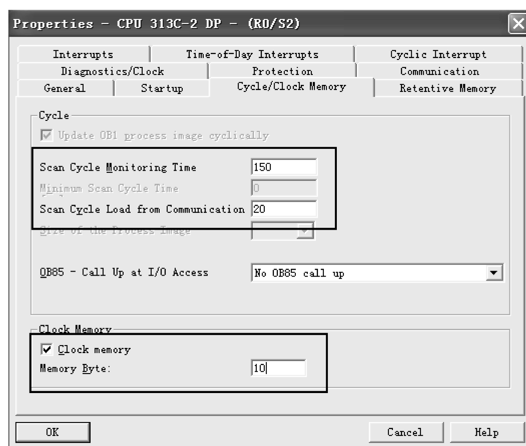



图 8-1　CPU 属性的“Cycle/Clock Memory”设置

系统已经在 OB1 中定义了一些局部变量，如表 8-3 所示。用户通过访问这些变量得到 OB1 的起动信息及其他的属性。

表 8-3　OB1 的局部变量

| 变　　量 | 类　　型 | 描　　述 |
|---|---|---|
| OB1_EV_CLASS | BYTE | 事件等级和标识码：B#16#11：OB1 激活 |
| OB1_SCAN_1 | BYTE | B#16#01：暖起动完成<br>B#16#02：热起动完成<br>B#16#03：主循环完成<br>B#16#04：冷起动完成<br>B#16#05：当前一个主站 CPU 停机，后备新主站 CPU 的第一次 OB1 循环 |
| OB1_PRIORITY | BYTE | 优先级 1 |
| OB1_OB_NUMBR | BYTE | OB 号（01） |
| OB1_ RESERVED_1 | BYTE | 备用 |
| OB1_ RESERVED_2 | BYTE | 备用 |
| OB1_PREV_CYCLE | INT | 上一次 OB1 的循环时间（ms） |
| OB1_MIN_CYCLE | INT | 自 CPU 起动，最短一次 OB1 的循环时间（ms） |
| OB1_MAX_CYCLE | INT | 自 CPU 起动，最长一次 OB1 的循环时间（ms） |
| OB1_DATE_TIME | DATE_AND_TIME | OB 被调用的日期和时间 |

## 8.3　日期时间中断组织块（OB10～OB17）

### 8.3.1　概述

STEP 7 提供多达 8 个日期时间中断组织块（OB10～OB17），这 8 个 OB 的默认优先级相同，都没有指定缺省时间。但是，只有 S7－400 系列的高级 CPU 才可以使用全部 8 个

OB，S7－300 系列中 CPU 318 能使用 OB10 和 OB11，其余 CPU 只能使用 OB10。下面以 OB10 为例来说明其用法。

OB10 可以运行一次或周期性地运行，但时间间隔有限制，只能是每分钟、每小时、每天、每周、每月和每月末等。在简单应用中，可以在 STEP 7 中通过给 CPU 设置参数实现；在较为复杂的应用中，可以运用系统功能 SFC28 ~ SFC31，并且编程实现。

系统已经在 OB10 中定义了一些局部变量，如表 8-4 所示，为用户编程提供了便捷。

表 8-4 OB10 的局部变量

| 变　　量 | 类　　型 | 描　　述 |
|---|---|---|
| OB10_EV_CLASS | BYTE | 事件级和识别码：B#16#11 = 中断激活 |
| OB10_STRT_INFO | BYTE | B#16#11 ~ B#16#18：分别是起动请求 OB10 ~ OB17 |
| OB10_PRIORITY | BYTE | 分配的优先级：默认 2 |
| OB10_OB_NUMBR | BYTE | OB 号（10 ~ 17） |
| OB10_RESERVED_1 | BYTE | 保留 |
| OB10_RESERVED_2 | BYTE | 保留 |
| OB10_PERIOD_EXE | WORD | OB 以特殊的间隔运行：<br>W#16#0000：一次<br>W#16#0201：每分钟一次<br>W#16#0401：每小时一次<br>W#16#1001：每天一次<br>W#16#1201：每周一次<br>W#16#1401：每月一次<br>W#16#1801：每年一次<br>W#16#2001：每月底一次 |
| OB10_DATE_TIME | DATE_AND_TIME | 调用 OB 时的日期和时间 |

参数“OB10_PRIORITY”是各个 OB 的优先级，默认为 2，若用户在系统中使用了多个日期时间中断组织块，则可以通过设置这个参数实现改变中断的优先级。可以通过改变参数“OB10_PERIOD_EXE”的值设置循环间隔；注意这些设置值是固定的，与循环间隔是一一对应的关系。

## 8.3.2 应用方法

在起动日期时间中断时，必须首先设置和激活中断。以下三种方式可以设置和激活中断：

① 自动起动日期时间中断。可以通过 STEP 7 设置并激活中断。具体方法如下：在 STEP7 的硬件组态窗口中，双击项目中机架 CPU 所在的行，打开 CPU 属性对话框，点击“Time-Of-Day Interrupts”选项页，设置框就显示出当前 CPU 可以使用的日期时间中断块，用户可以选中“Active”选择框激活 OB10。在“Execution”列表框内选择循环时间间隔，并在其后的两个编辑框内输入起动中断的日期和时间。设置和激活日期时间中断如图 8-2 所示。保存后下载硬件组态，就实现了日期时间中断的自动起动。

② 可以在 STEP 7 中设置日期时间中断，但不激活“Active”选择框。然后通过程序调用 SFC30“ACT-TINT”来激活日期时间中断。

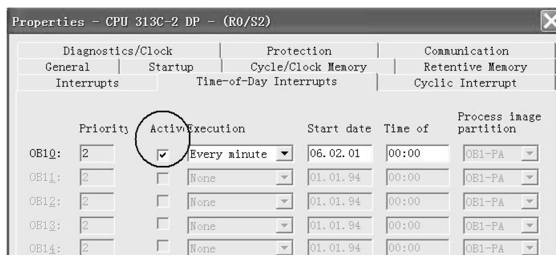



图 8-2　设置和激活日期时间中断

③ 可以通过调用 SFC28“SET_TINT”设置日期时间中断，再通过调用 SFC30“ACT_TINT”，激活日期时间中断。具体方法参见例 1。

用同样的方法也可以在 STEP 7 中取消日期时间中断。在程序中需要时可通过调用 SFC29“CAN_TINT”取消日期时间中断，调用 SFC31“QRY_TINT”查询日期时间中断。如表 8-5 所示为 SFC28 ~ SFC31 的参数说明。

**表 8-5　SFC28 ~ SFC31 的参数说明**

| 参数 | 声明 | 数据类型 | 存储区域 | 参数说明 |
|---|---|---|---|---|
| OB_NR | INPUT | INT | I，Q，M，D，L，常数 | 中断 OB 号（OB10 ~ OB17） |
| SDT | INPUT | DT | D，L，常数 | 起动时间：所定义的起动时间中秒和毫秒省略，用 0 代替 |
| PERIOD | INPUT | WORD | I，Q，M，D，L，常数 | STD 起动周期<br>W#16#0000：一次<br>W#16#0201：每分钟<br>W#16#0401：每小时<br>W#16#1001：每天<br>W#16#1201：每周<br>W#16#1401：每月<br>W#16#1801：每年<br>W#16#2001：每月末 |
| RET_VAL | OUTPUT | INT | I，Q，M，D，L | 如故障发生，返回值包含故障代码 |
| STATUS | OUTPUT | WORD | I，Q，M，D，L | 日期时间中断状态 |

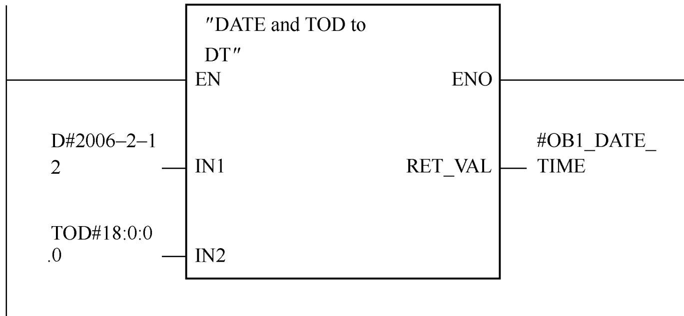
## 8.3.3　应用实例

**例 1**　自 2006－2－12 的 18 点整开始，每分钟中断一次，每次中断使 MW0 自动加 1。要求用 I0.0 的上升沿脉冲设置和起动日期时间中断 OB10，用 I0.1 的高电平禁止日期时间中断 OB10。

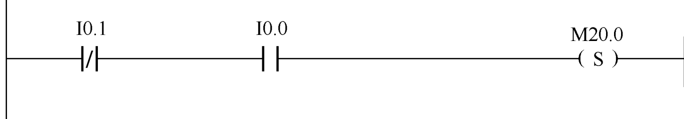
如图 8-3 所示为主程序 OB1，如图 8-4 所示为中断程序 OB10。

说明：在 OB1 的 Network1 中调用系统 IEC 功能 FC3“DATE and TOD to DT”，将日期（格式为 DATE）和时间（格式为 TOD）数据合并，并且转换为 DATE_AND_TIME 格式（简称 DT）的数据，并且暂时置于局部变量 OB1_DATE_TIME。因为在 SFC28“SET_ TINT”功能块中，输入参数 SDT（设置中断的起动起始时间）的数据类型为 DT 格式，所以必须进行数据类型的合并与转换。在 Network4 中，“OB1_ DATE_ TIME”作为 SFC28 的输入参数 SDT，W#16#0201 表示每分钟中断一次（见表 8-5）。在 OB10 中，只需将 MW0 自动加 1 即可，表明调用 OB10 的次数。

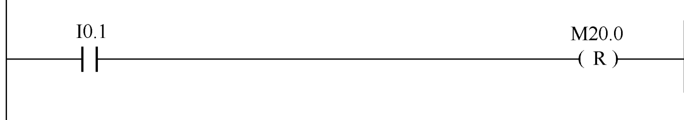
Network 1: 合并日期时间项



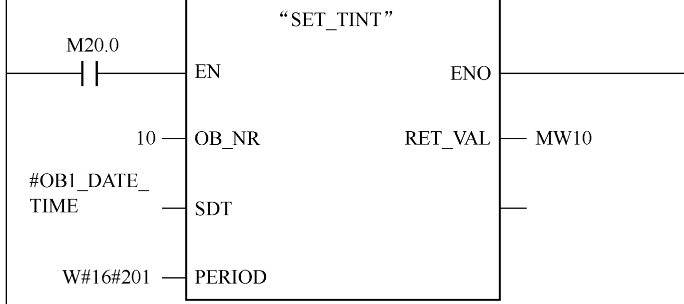

Network 2: 保持设置与激活脉冲



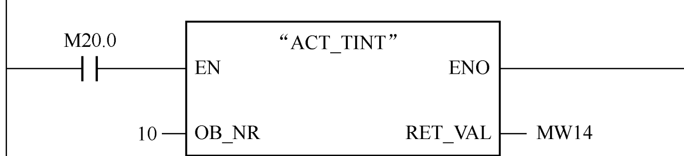

Network 3: 清除设置与激活脉冲




Network 4: 设置日期时间中断




Network 5: 激活日期时间中断




Network 6: 取消日期时间中断

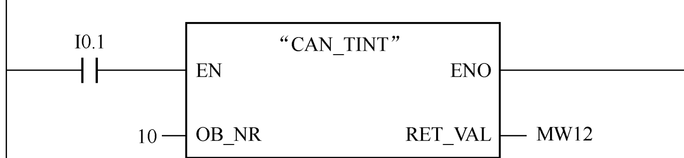



图 8-3　主程序 OB1

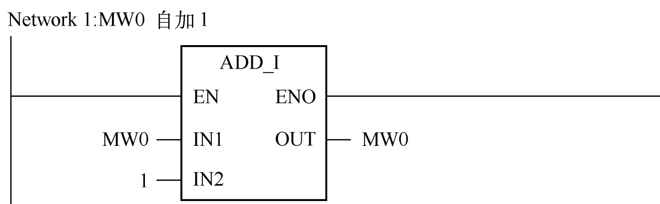



图 8-4　中断程序 OB10

程序保存好后就可以下载到实际 PLC 或 PLCSIM 仿真软件中了。需要注意的是，一定要把所有的程序块都下载，包括 FC3、SFC28 等。选中将要下载的程序块，如图 8-5 所示，再点击工具栏的下载按钮即可。

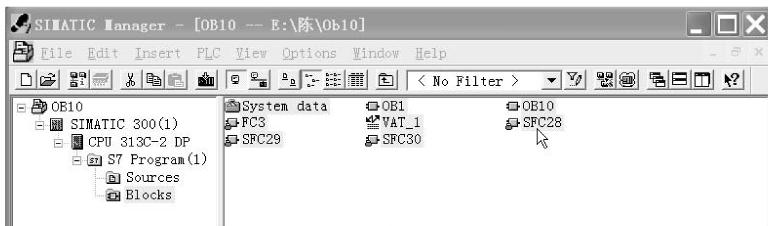
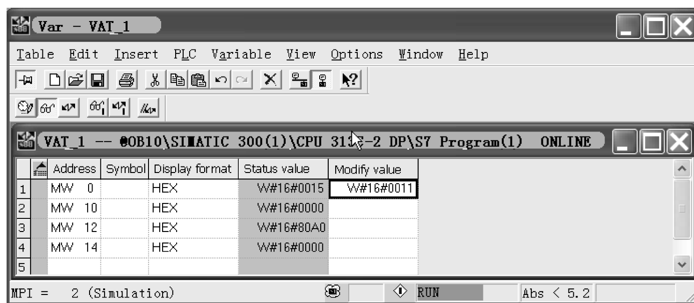



图 8-5　下载程序块

为了方便监控程序运行情况，在 STEP 7 的 Blocks 中插入变量表，然后打开，填入地址 MW0、MW10、MW12 等，并点击工具栏中按钮，利用变量表监控程序的运行，如图 8-6 所示。在 Network2 中，用 I0.0 设置并激活 OB10，可以观察到 MW0 每隔一分钟便自动加 1；在 Network3 中，用 I0.1 取消此中断。MW10、MW12 和 MW14 分别是 SFC28、SFC29 和 SFC30 的返回值，若有故障产生，将会显示相应的故障代码。具体故障代码的含义请参考系统手册《SIMATIC S7 -300/400 的系统软件和标准功能》，以后涉及的故障代码均参考此手册。




| | Address | Symbol | Display format | Status value | Modify value |
|---|---|---|---|---|---|
| 1 | MW 0 | | HEX | W#16#0015 | W#16#0011 |
| 2 | MW 10 | | HEX | W#16#0000 | |
| 3 | MW 12 | | HEX | W#16#80A0 | |
| 4 | MW 14 | | HEX | W#16#0000 | |
| 5 | | | | | |

图 8-6　利用变量表监控程序的运行

## 8.4　延时中断组织块（OB20 ~ OB23）

### 8.4.1　概述

在延时中断组织块中，用户可以编写将要延时执行的程序。PLC 中定时器的定时时间与

扫描方式有关，其精度受不断变化的循环周期的影响，而使用延时中断组织块则可以提高延时的精度，延时时间分辨率为 1 ms。

S7 提供了 4 个延时中断组织块（OB20 ~ OB23），S7 - 300 系列中 CPU 318 能使用 OB20 和 OB21，其余 S7 - 300CPU 只能使用 OB20，S7 - 400 系列 CPU 可以使用的延时中断 OB 与其型号有关。我们以 OB20 为例来说明其用法。

如果以下任一情况发生，操作系统将调用一个异步错误组织块：

① 在调用 SFC32 时，OB20 并没有下载到 CPU 中。

② 一个延时中断的执行还没结束，下一个延时中断又被起动。

OB20 的局部变量如表 8-6 所示，这些变量的定义为用户编程提供了方便。其中变量 "OB20_PRIORITY" 是代表 OB20 的优先级，默认为 3，可以通过设置这个变量参数改变优先级。

表 8-6　OB20 的局部变量

| 变　　量 | 类　　型 | 描　　述 |
|---|---|---|
| OB20_EV_CLASS | BYTE | 事件级别和识别码，B#16#11：中断激活 |
| OB20_STRT_INF | BYTE | B#16#20 ~ B#16#21：OB20 ~ OB23 的起动请求 |
| OB20_PRIORITY | BYTE | 分配的优先级：默认值为 3（OB20） ~6（OB23） |
| OB20_OB_NUMBR | BYTE | OB 号（20 ~23） |
| OB20_RESERVED_1 | BYTE | 保留 |
| OB20_SIGN | WORD | 用户 ID：SFC32 的输入参数 SIGN |
| OB20_DTIME | TIME | 以毫秒形式组态的延时时间 |
| OB20_DATE_TIME | DATE_AND_TIME | OB 被调用时的日期和时间 |

## 8.4.2　应用方法

首先可以在 STEP 7 中查看可支持的延时中断 OB。具体方法是：在 STEP 7 的硬件组态窗口中，双击项目中机架上 CPU 所在的行，打开 CPU 属性对话框，点击 "Interrupts" 选项页，设置框中显示出当前 CPU 支持的延时中断组织块，如图 8-7 所示。

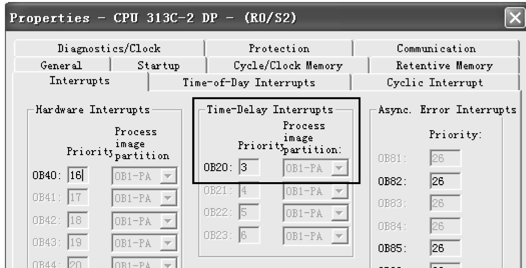



图 8-7　CPU 支持的延时中断组织块

延时中断组织块 OB20 经过一段指定的延时时间后运行。在程序中，系统提供了 SFC 32 ~ SFC 34 三个系统功能块供用户编程使用。OB20 在调用 SFC32 "SRT_DINT" 后起动，延时时间在 SFC32 的参数中设定；如果在延迟时间未到之前想取消延时程序的执行，可以调用 SFC33

"CAN_DINT"；同时可以使用 SFC34"QRY_DINT"查询延迟中断的状态。在应用中，必须将 OB20 和使用的 SFC 都下载到 PLC 中。如表 8-7 所示为 SFC32～SFC34 的参数说明。

表 8-7　SFC32～SFC34 的参数说明

| 参数 | 声明 | 数据类型 | 存储区域 | 参 数 说 明 |
|---|---|---|---|---|
| OB_NR | INPUT | INT | I，Q，M，D，L，常数 | OB 号（OB20～OB23），延时后被起动 |
| DTIME | INPUT | TIME | I，Q，M，D，L，常数 | 延时值（1～60000 ms） |
| SIGN | INPUT | WORD | I，Q，M，D，L，常数 | 当延时中断 OB 被调用时，在起始事件信息中出现的开始标志 |
| RET_VAL | OUTPUT | INT | I，Q，M，D，L | 如故障发生，返回值包含故障代码 |
| STATUS | OUTPUT | WORD | I，Q，M，D，L | 时间中断的状态 |

## 8.4.3　应用实例

**例 2**　M20.0 控制起动延时中断 OB20，延时 10s 后中断一次，中断程序使 MW0 自动加 1，M20.1 控制取消延时中断 OB20。

如图 8-8 所示是主程序 OB1，延时中断程序与例 1 相同。

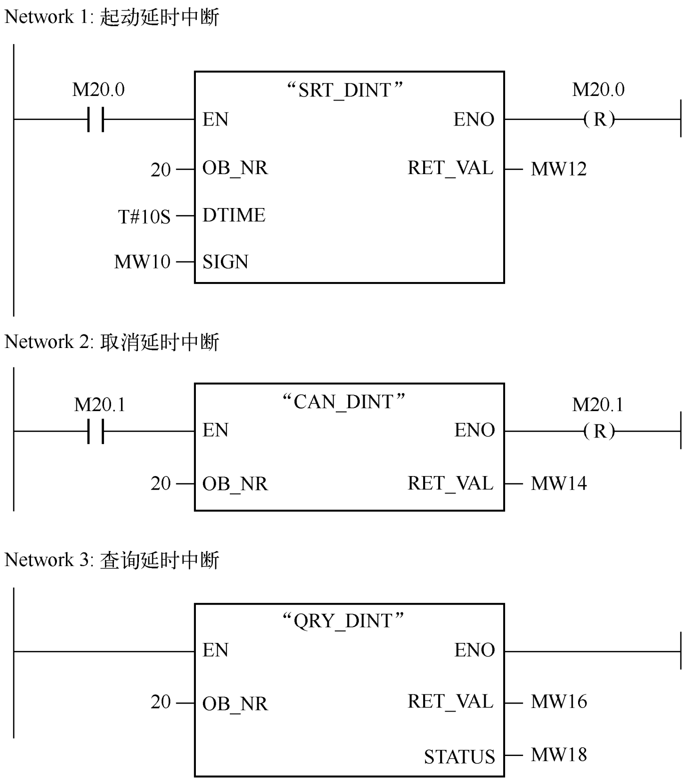



图 8-8　主程序 OB1

说明：在 OB1 的 Network1 中调用系统功能 SFC32 起动延时中断，DTIME 端是延时的时间设置，此时为 T#10s。程序编译保存好后就可以下载到实际 PLC 或 PLCSIM 仿真软件中了。需要注意的是，一定要把所有的程序块都下载，包括 OB20、SFC32 等，选中要下载的

程序块，再点击工具栏的下载按钮即可。

为了方便监控程序运行情况，插入变量表，填入存储器地址 MW0、M20.0、M20.1 等，并点击工具栏中按钮，利用变量表监控程序的运行如图 8-9 所示。此时可以监控 MW0 的变化，在变量表中使用修改变量（Modify Variable）按钮强制使 M20.0 为 True 起动 OB20，10s 后延迟时间到，MW0 自动加 1。再将 M20.0 置为 true，10s 后延迟时间到，MW0 再加 1。如果当延迟时间未到，此时将 M20.1 置为 true，那么这次时间延迟中断被取消，MW0 不会加 1，每次执行的状态都可以从 MW18 中读出，MW12、MW14 和 MW16 分别是 SFC32、SFC33 和 SFC34 的返回值，若有故障产生，将会显示相应的故障代码。

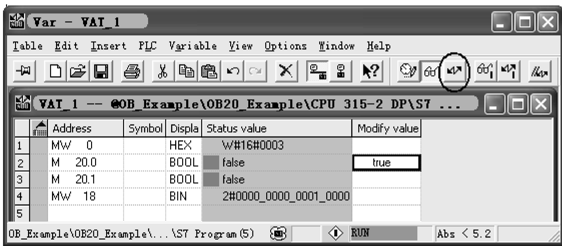



图 8-9　利用变量表监控程序的运行

## 8.5　循环中断组织块（OB30～OB38）

### 8.5.1　概述

STEP 7 提供了 9 个循环中断组织块（OB30 ～OB38），它们经过一段固定的时间间隔中断用户的程序。S7 -300 系列中 CPU 318 能使用 OB32 和 OB35，其余 S7 -300CPU 只能使用 OB35，S7 -400 系列 CPU 可以使用的循环中断组织块与其型号有关。如表 8-8 所示为每个循环中断组织块默认的时间间隔和优先级，用户可以设置自己需要的时间间隔和优先级。我们以 OB35 为例来说明其用法。

**表 8-8　循环中断组织块默认的时间间隔和优先级**

| OB 号 | 默认的时间间隔 | 默认的优先级 |
|---|---|---|
| OB30 | 5 s | 7 |
| OB31 | 2 s | 8 |
| OB32 | 1 s | 9 |
| OB33 | 500 ms | 10 |
| OB34 | 200 ms | 11 |
| OB35 | 100 ms | 12 |
| OB36 | 50 ms | 13 |
| OB37 | 20 ms | 14 |
| OB38 | 10 ms | 15 |

循环中断组织块 OB35 是按设定的时间间隔循环执行的中断程序，间隔时间从 STOP 切换到 RUN 模式时开始计算。用户可以在 OB35 中周期的调用 PID 模块（FB41/42/43），完成 PID 调节，也可以在 OB35 中调用周期的数据发送指令，完成数据发送功能等。

如表 8-9 所示为 OB35 的局部变量。

表 8-9　OB35 的局部变量

| 变　量 | 类　型 | 描　述 |
|---|---|---|
| OB35_EV_CLASS | BYTE | 事件级别和识别码 B#16#11：中断激活 |
| OB35_STRT_INF | BYTE | B#16#30：循环中断组织块 OB 的起动请求（只有 H 型 CPU 并且明确地为其组态）B#16#31 ~ B#16#39：OB30 ~ OB38 的起动请求 |
| OB35_PRIORITY | BYTE | 分配的优先级：默认 7（OB30） ~15（OB38） |
| OB35_OB_NUMBR | BYTE | OB 号（30 ~38） |
| OB35_RESERVED_1 | BYTE | 保留 |
| OB35_RESERVED_2 | BYTE | 保留 |
| OB35_PHASE_OFFSET | WORD | 相位偏移量［ms］ |
| OB35_EXC_FREQ | INT | 时间间隔，以 ms 计 |
| OB35_DATE_TIME | DATE_AND_TIME | OB 调用时的日期和时间 |

## 8.5.2　应用方法

首先可以在 STEP 7 中查看可支持的循环中断 OB。具体方法是：在 STEP 7 的硬件组态窗口中，双击项目中机架上 CPU 所在的行，打开 CPU 属性对话框，点击“Cyclic Interrupts”选项页，设置框就显示出当前 CPU 可以使用的循环中断块，可以设置循环中断如图 8-10 所示。用户可以在“Priority”编辑框中设置当前循环 OB 的优先级，在“Execution”编辑框中可改变默认的间隔时间，范围是 0 ~ 60000 ms。

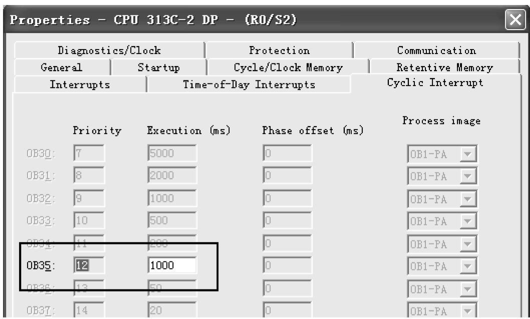



图 8-10　设置循环中断

循环中断组织块每一次运行的时间一定要短于中断的间隔。如果一个循环中断组织块没有执行完，循环中断时间到，又要求循环中断组织块运行，则时间故障组织块 OB80 起动。反复的循环中断导致了故障程序的运行。

与 OB20 使用方法不同的是，系统没有提供专用的激活和禁止循环中断 SFC，但可以运

用 SFC39～SFC42 取消、延时和再次使能循环中断。SFC40“EN_INT”是用于激活新的中断和异步错误的系统功能，其参数 MODE 为 0 时激活所有的中断和异步错误；为 1 时激活部分中断和异步错误，为 2 时激活指定的 OB 编号对应的中断和异步错误。SFC39“DIS_INT”是禁止新的中断和异步错误的系统功能，其参数 MODE 为 2 时禁止指定的 OB 编号对应的中断和异步错误，MODE 参数有多种选择，其他可查阅系统手册《SIMATIC S7－300/400 的系统软件和标准功能》。以上两个 SFC 的 MODE 参数必须用十六进制数来设置。如表 8-10 所示为 SFC39～SFC42 的参数说明。

表 8-10　SFC39～SFC42 的参数说明

| 参数 | 声明 | 数据类型 | 存储区域 | 参 数 说 明 |
|---|---|---|---|---|
| OB_NR | INPUT | INT | I，Q，M，D，L，常数 | OB 号（OB30－OB38） |
| MODE | INPUT | BYTE | I，Q，M，D，L，常数 | 定义被禁止的中断和异步故障 |
| RET_VAL | OUTPUT | INT | I，Q，M，D，L | 如故障发生，返回值包含故障代码<br>在 SFC41 中：延迟的编号（等于 SFC41 调用的编号）<br>在 SFC42 中：激活报警中断调用 SFC 的次数或者故障信息 |

## 8.5.3　应用实例

**例 3**　每 3 min 中断一次，每次中断使 MW0 自动加 1。要求用 I0.0 的上升沿脉冲设置和起动循环中断 OB35，用 I0.1 的高电平禁止日期时间中断 OB10。

首先在图 8-10 中设置循环间隔时间为 3000 ms，表示每 3 s 调用 OB35 一次。

如图 8-11 所示是主程序 OB1，循环中断程序与例 1 相同。

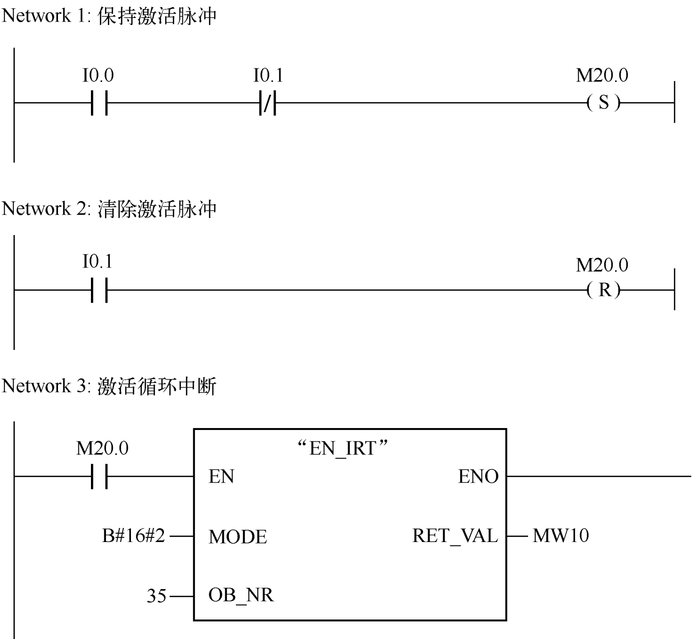



图 8-11　主程序 OB1

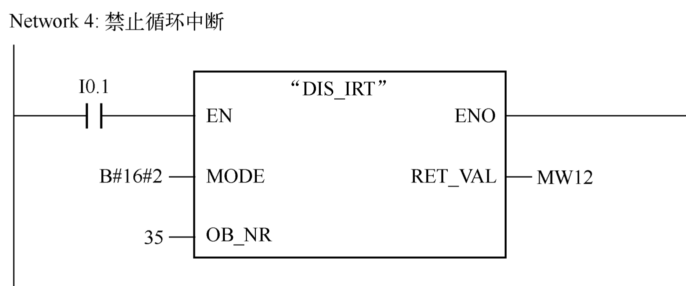



图 8-11　主程序 OB1（续）

说明：在 OB1 的 Network3 和 Network4 中 MODE 为 B#16#2 分别表示激活指定的 OB35 所对应的循环中断和禁止新的循环中断。

程序保存好后就可以下载到实际 PLC 或 PLCSIM 仿真软件中了。为了方便监控程序运行情况，在 STEP 7 的 Blocks 中插入变量表，方法同例 1。程序进入 RUN 模式后，可以观察到每 3s，MW0 自动加 1。当产生 I0. 1 高电平时，循环中断被禁止，MW0 停止自加 1；当输入 I0. 0 脉冲时，循环中断又被激活，MW0 又开始自动加 1。

## 8. 6　硬件中断组织块（OB40 ~ OB47）

### 8. 6. 1　概述

硬件中断组织块（OB40 ~ OB47）用于快速响应信号模块 SM、通信处理模块 CP 和功能模块 FM 的信号变化。例如：当数字量输入模块的一个通道有上升沿信号来时，若该模块有中断能力，则触发一个硬件中断，中断服务程序就置于硬件中断组织块中。

S7 提供多达 8 个独立的硬件中断组织块（OB40 ~ OB47）。S7 －300 系列中 CPU 318 能使用 OB40 和 OB41，其余 S7 －300 系列 CPU 只能使用 OB40，S7 －400 系列 CPU 可以使用的硬件中断组织块与其相应的型号有关。如表 8-11 所示描述了 OB40 的局部变量。

表 8-11　OB40 的局部变量

| 变　　量 | 类　　型 | 描　　述 |
|---|---|---|
| OB40_EV_CLASS | BYTE | 事件级别和诊断号：<br>B#16#11：中断被激活 |
| OB40_STRT_INF | BYTE | B#16#41：中断通过中断行 1<br>B#16#42 ~ B#16#44：<br>中断通过中断行 2 ~ 4（只对 S7 －400）<br>B#16#45：WinAC 通过 PC 触发的中断 |
| OB40_PRIORITY | BYTE | 分配优先级：默认 16（OB40） ~23（OB47） |
| OB40_OB_NUMBR | BYTE | OB 号（40 ~ 47） |
| OB40_RESERVED_1 | BYTE | 保留 |
| OB40_IO_FLAG | BYTE | 输入模板：B#16#54 输出模板：B#16#55 |
| OB40_MDL_ADDR | WORD | 触发中断模块的逻辑地址 |

（续）

| 变　　量 | 类　　型 | 描　　述 |
|---|---|---|
| OB40_POINT_ADDR | DWORD | 数字模板：带有模板输入状态的位字段（0 位对应第一个输入）<br>模拟模板：带有限幅信息输入通道的位字段<br>CP 或 IM：模块中断状态（不是与用户相关的） |
| OB40_DATE_TIME | DATE_AND_TIME | 被调用的日期和时间 |

## 8.6.2　应用方法

首先可以在 STEP 7 中查看可支持的硬件中断组织块。具体方法是：在 STEP 7 的硬件组态窗口中，双击项目中机架上 CPU 所在的行，打开 CPU 属性对话框，点击“Interrupts”选项页，可以看到 CPU 支持的硬件中断块，如图 8-12 所示。在此也可以为硬件中断 OB 选择优先级。

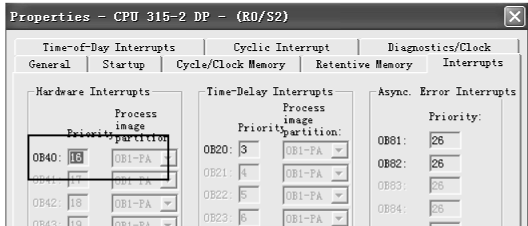



图 8-12　CPU 支持的硬件中断块

通过 STEP 7 进行参数赋值，可以为能够触发硬件中断的每一个信号模板指定参数。

## 8.6.3　应用实例

**例 4**　I0.0 的上升沿作为硬件中断触发脉冲，使用硬件中断 OB40，当来一次 I0.0 的上升沿，就使 MW0 自动加 1。

首先在硬件组态中设置中断触发信号。如上所述，并不是所有的信号模块都具有中断功能。此例中，我们需要一个数字量输入模块，如图 8-13 所示为硬件组态，其右视图硬件目录的“DI－300”中有此版本软件支持的所有 SM321，单击一个模块后，右下角处将出现这个模块的基本信息。然后插入 CPU 315－2DP 和一块具有中断功能的数字量输入模板（如 SM321，订货号 6ES7 321－7BH01－0AB0）。双击模板，选择“Inputs”选项，同时激活“Hardware interrupt ”和“Trigger for Hardware Interrupt”选项，如图 8-14 所示设置数字量输入模板的中断。

说明：在此例中，也可以按照例 3 的方法，用 SFC39 和 SFC40 来取消和激活中断。在此，我们只设置中断模块，并在 OB40 中编程即可完成功能。如图 8-15 所示为硬件中断程序 OB40。在 Network2 中利用局部变量 OB40_MDL_ADDR 和 OB40_POINT_ADDR，在 MW10 和 MD12 中得到输入模块的起始地址和产生的中断号。

我们使用了两个 OB40 的局部变量 OB40_MDL_ADDR 和 B40_POINT_ADDR，用于观察中断是由哪个模块的哪个通道产生的。利用变量表监控程序的运行如图 8-16 所示。MW0 当前值为 000D，它自动加 1 已经是 13 了，表示已经中断了 13 次；MW10 为 0000，表示这个

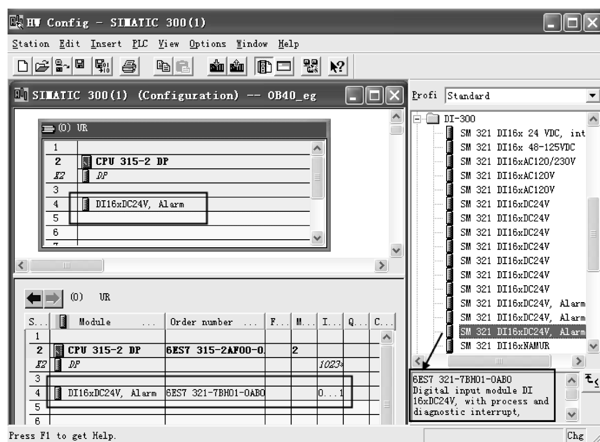



图 8-13　硬件组态

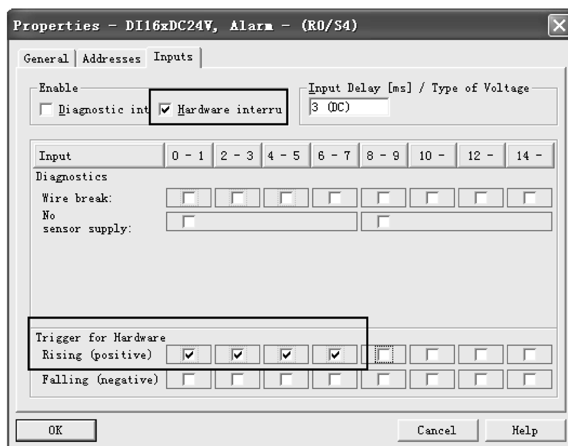
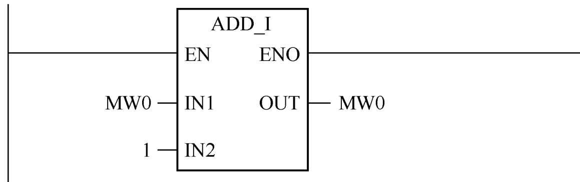



图 8-14　设置数字量输入模板的中断

OB40:“Hardware Interrupt”

Network 1:MW0 自加 1




Network 2:MW10 是输入模块的起始地址，MD12 是产生中断的通道号

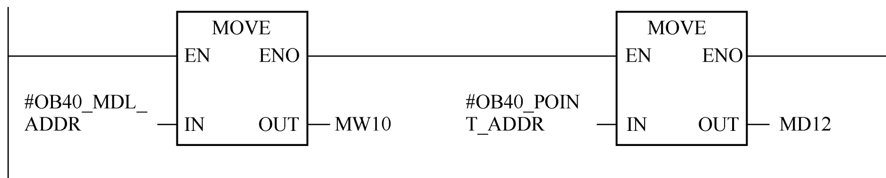



图 8-15　硬件中断程序 OB40

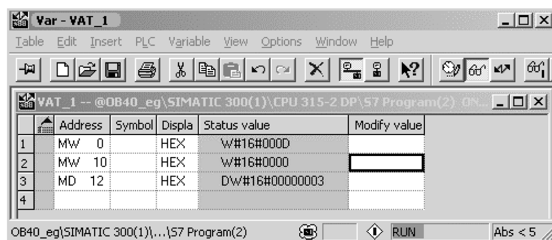


图 8-16　利用变量表监控程序的运行

硬件中断由起始地址为 0 的模块产生；MD12 为 3，表示由第 3 个通道产生，即 I0.3 的上升沿产生硬件中断。当然也可使用这个模块的其他通道，但是必须在如图 8-14 所示组态时激活这些通道。

若使用实际 PLC 模拟硬件中断，只要在对应模块的输入中给上升沿脉冲即可。下面详细介绍在 PLCSIM 仿真软件中模拟硬件中断的方法。硬件组态和软件程序下载到 PLCSIM 中后，将 PLC 的状态切换到 RUN 或 RUN－P 模式，用鼠标模拟产生 I0.3 的上升沿脉冲的方法是：选择“Execute/Trigger Error OB/Hardware Interrupt（OB40－OB47）…”，打开 Hardware Interrupt OB 对话框，如图 8-17 所示。文本框“Module address”和“Module status”分别对应 OB40 的局部变量 OB40_MDL_ADDR 和 OB40_POINT_ADDR，在这两个编辑框中分别输入 0 和 3，再单击“Apply”按钮，就触发了指定通道（模块起始地址为 0，通道号为 3）的硬件中断，系统立即执行一次 OB40，同时在“Interrupt”显示框内将自动显示出当前执行的硬件中断 OB 的编号 40。可以修改两个编辑框，再单击“Apply”按钮，就又一次触发了指定通道的硬件中断。单击“OK”按钮与单击“Apply”按钮作用相同，并退出对话框。

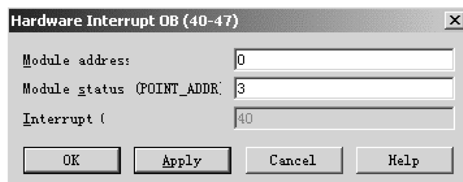



图 8-17　Hardware Interrupt OB 对话框

## 8.7　异步错误组织块

西门子的大中型 PLC 具有很强的错误和故障的检测及处理能力。当 CPU 检测到某种错误后，操作系统将自动调用对应的组织块，用户可以事先在对应的组织块中编写程序，就可以对发生的错误或故障采取相应的措施。如果用户没有建立相应的组织块，那么发生错误或故障后，CPU 将转为 STOP 模式。

像所有的组织块一样，错误处理组织块中包含了关于调用原因的附加信息。操作系统将这些信息记录在该组织块的局部变量中，用户可以在程序中对它们进行访问，以便于故障诊断。当 CPU 不支持某些错误组织块时，相关的错误信息就不会记录在 OB 中。

为避免发生某种错误时 CPU 进入停止状态，可以在 CPU 中建立一个相应的空错误组织

块，也可以在错误组织块中编程实现所希望的响应。如果需要，在执行完规定指令后，调用系统功能 SFC 46 申请停机。

能被 CPU 检测到并且用户可以通过组织块对其进行处理的错误或故障分为两种基本类型：异步错误和同步错误。异步错误组织块是处理与 PLC 硬件或操作系统密切相关的错误，这类错误与程序执行无关。同步错误组织块是处理与程序执行有关的错误。

具有诊断能力的 CPU 支持异步错误组织块，如表 8-12 所示为调用异步错误组织块的典型案例。另外，H 系列的 CPU 还支持 OB70、OB72 和 OB73。

**表 8-12　调用异步错误组织块的典型案例**

| 错 误 类 型 | 案　　例 |
| --- | --- |
| 时间错误 OB80 | 超出最大循环扫描时间 |
| 电源故障 OB81 | 后备电池失效 |
| 诊断中断 OB82 | 有诊断能力模块的输入断线 |
| 插入/移除中断 OB83 | 在运行时移除 S7 - 400 的信号模块 |
| CPU 硬件故障 OB84 | MPI 接口上出现错误的信号电平 |
| 程序执行错误 OB85 | 更新映像区错误（模块有缺陷） |
| 机架错误 OB86 | 扩展设备或 DP 从站故障 |
| 通信错误 OB87 | 读取信息格式错误 |

### 8.7.1　时间错误处理组织块（OB80）

OB 执行时出现故障，操作系统调用 OB80。这样的故障包括：循环时间超出、执行 OB 时应答故障、向前移动时间以至于跃过了 OB 的起动时间等。例如，当循环中断组织块仍在执行前一次调用时，该组织块的起动事件发生，操作系统调用 OB80。如果 OB80 未编程，CPU 变为 STOP 方式。

循环监控时间的默认值为 150 ms，时间错误包括实际循环时间超过设置的循环时间，这可能由于向前修改时间而跳过日期时间中断、处理优先级时延迟太多等情况引起。为 OB80 编程时应判断是哪个日期时间中断被跳过，使用 SFC29“CAN_TINT”可以取消被跳过的日期时间中断。

### 8.7.2　电源故障处理组织块（OB81）

电源故障包括后备电池失效或未安装电池，电源故障出现和消失时操作系统都要调用 OB81。如果 OB81 未编程，CPU 并不转换为 STOP 方式。用户可以用 SFC39 ~ SFC42 来禁用、延时或再使能电源故障组织块。

### 8.7.3　诊断中断组织块（OB82）

诊断中断组织块 OB82 用于检查和评估诊断中断。如果模块（包括 CPU 或其他模块）具有诊断能力而且激活了诊断中断，当它检测到故障时，输出一个诊断中断请求给 CPU（到来和离去事件），操作系统将调用 OB82。故障包括：有诊断功能的模块的断线故障，模拟输入模块的电源故障，输入信号超过模拟量模块的测量范围等。如表 8-13 所示为 OB82

的局部变量，用户可以利用这些变量得到一些诊断信息。

表 8-13　诊断中断 OB82 的局部变量

| 变　量 | 类　型 | 描　述 |
|---|---|---|
| OB82_EV_CLASS | BYTE | 事件级别和标识：B#16#38：离去事件<br>B#16#39：到来事件 |
| OB82_FLT_ID | BYTE | 故障代码（B#16#42） |
| OB82_PRIORITY | BYTE | 优先级；可通过 STEP 7 选择（硬件组态） |
| OB82_OB_NUMBR | BYTE | OB 号（82） |
| OB82_IO_FLAG | BYTE | 输入模板：B#16#54<br>输出模板：B#16#55 |
| OB82_MDL_ADDR | WORD | 故障发生处模板的逻辑起始地址 |
| OB82_MDL_DEFECT | BOOL | 模板故障 |
| OB82_INT_FAULT | BOOL | 内部故障 |
| OB82_EXT_FAULT | BOOL | 外部故障 |
| OB82_PNT_INFO | BOOL | 通道故障 |
| OB82_EXT_VOLTAGE | BOOL | 外部电压故障 |
| OB82_FLD_CONNCTR | BOOL | 前连接器未插入 |
| OB82_NO_CONFIG | BOOL | 模板未组态 |
| OB82_CONFIG_ERR | BOOL | 模板参数不正确 |
| OB82_MDL_TYPE | BYTE | 位 0 至 3：模板级别<br>位 4：通道信息存在<br>位 5：用户信息存在<br>位 6：来自替代的诊断中断 |
| OB82_SUB_MDL_ERR | BOOL | 子模板丢失或有故障 |
| OB82_COMM_FAULT | BOOL | 通信问题 |
| OB82_MDL_STOP | BOOL | 操作方式（0：RUN，1：STOP） |
| OB82_INT_PS_FLT | BOOL | 内部电源故障 |
| OB82_PRIM_BATT_FLT | BOOL | 电池故障 |
| OB82_BCKUP_BATT_FLT | BOOL | 全部后备电池故障 |
| OB82_RACK_FLT | BOOL | 扩展机架故障 |
| OB82_RAM_FLT RAM | BOOL | 故障 |

如果 OB82 未编程，那么 CPU 变为 STOP 方式。可以用 SFC39～SFC42 来禁止或延时并再使能诊断中断组织块，也可以使用 SFC51 “RDSYSST” 读出模块的诊断数据，使用 SFC 52 “WR_USMSG” 将这些信息存入诊断缓冲区，具体的参数说明请参考《SIMATIC S7－300/400 的系统软件和标准功能》。

**例 5**　如图 8-18 所示，液位传感器接入模拟量输入模块，利用带有诊断中断的模拟量模块实现当模块通道上的测量值超限时，中断 OB82 被调用，输出 Q4.1 就得电；当测量值回到允许范围内时，OB82 又将调用一次，输出 Q4.1 失电。

首先进行 PLC 的硬件组态，如图 8-19 所示。双击模拟量输入模块“AI8×12Bit”，将

出现该模块的参数设置对话框，点击“Input”选项页，如图8-20所示。在“Enable”选项框中选中“Diagnostic Interrupt”和“Hardware Interrupt When Limit Exceed（当超限时硬件中断）”，在0和1通道组中选中“Group Diagnostic”，在“Measuring”选项框中设置0－1通道组为“4DMU”（4线式电流传感器）、“4..20mA”，在“Trigger for Hardware”选项框中设置通道0的上限值为16mA。点击“OK”按钮确定。保存PLC的硬件组态配置并下载。

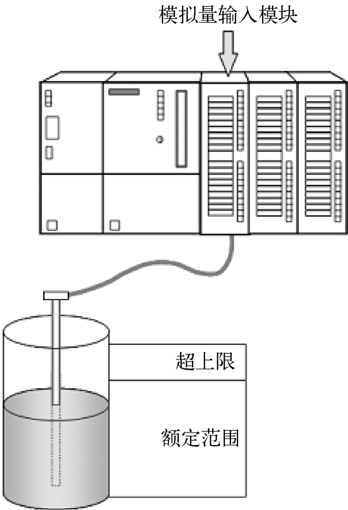


图8-18　液位传感器

说明：在CPU的“Blocks”中插入新组织块OB82，新建一个局部变量“incoming_outgoing”，诊断中断程序OB82如图8-21所示。当模拟量模块的值超过上限时，操作系统调用OB82，在Network1中，将OB82_MDL_ADDR中故障发生处模块的逻辑起始地址的值赋给局部变量“incoming_ outgoing”中；在Network2中，当中断事件到来（标识为B#16#39，即十进制数57），且发生故障模块的逻辑起始地址是256时，输出Q.1得电；反之当中断事件离去（标识为B#16#38，即十进制数56），且逻辑起始地址是256时，输出Q4.1失电。

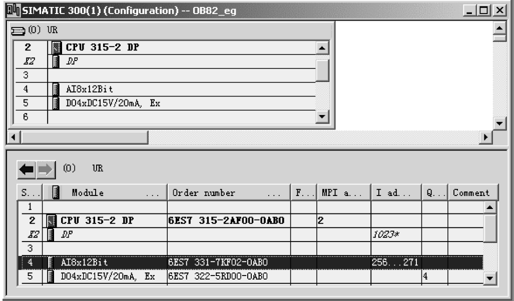


图8-19　硬件组态

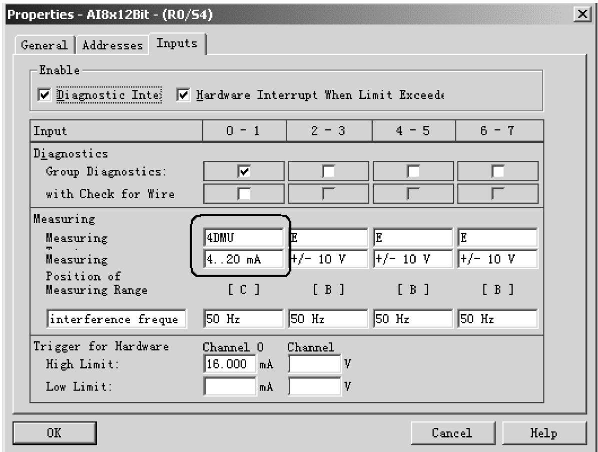
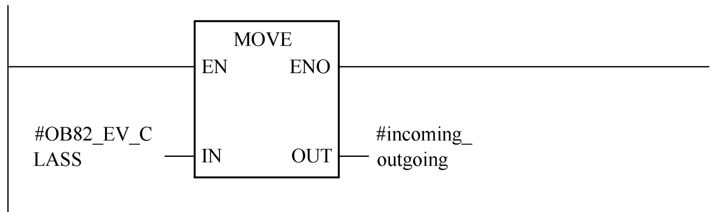


图8-20　模块参数设置“Input”选项页

OB82:“I/O Point Fault”

Network 1: 将“#OB82_EV_CLASS”赋值给“#incoming_outgoing”变量




Network 2: 当中断事件到来，且中断如地址是 256，输出 Q.1 得电 ；反之 Q4.1 失电。

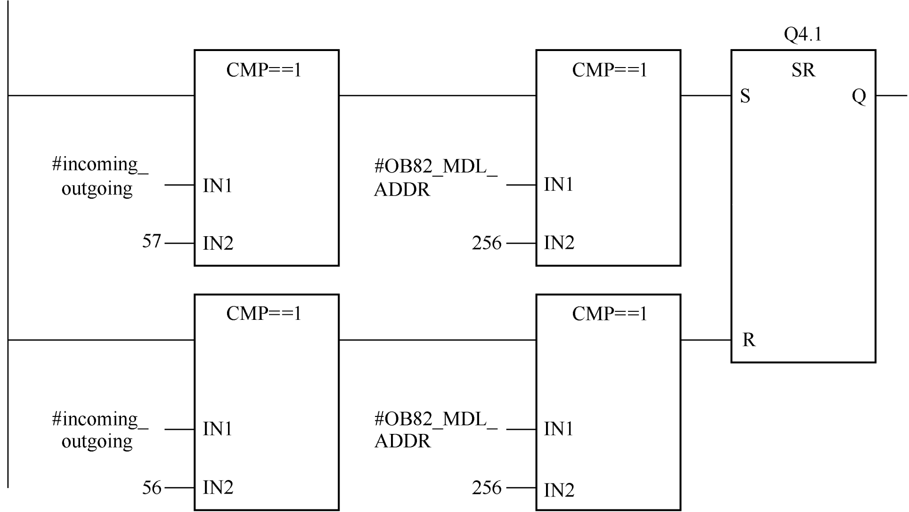



图 8-21　诊断中断程序 OB82

## 8.7.4　机架故障组织块（OB86）

当扩展机架（不是 CPU 318）、DP 主站系统、或分布式 I/O 中从站故障时（到来和离去事件），CPU 操作系统将调用 OB86。如果 OB86 未编程，当检测到这类故障时，CPU 进入 STOP 方式。

可以使用 SFC 39 ~ SFC42 来禁止、延时或使能 OB86。当在通信发生问题后或者访问不到配置的机架或从站时执行 OB86 程序，此时程序还可能需要调用 OB82 和 OB122 等组织块，当 OB86 执行时可以通过它的局部变量读出产生故障的错误代码和事件类型，通过它们的组合可以得出具体的错误信息，这些信息可以通过 OB86 的在线帮助查到，同时也可以读到产生错误的模块地址和机架信息，如表 8-14 所示描述了 OB86 的局部变量。

**表 8-14　OB86 的局部变量**

| 变　　量 | 类　　型 | 描　　述 |
|---|---|---|
| OB86_EV_CLASS | BYTE | 事件级别和标识：<br>B#16#38：离去事件<br>B#16#39：到来事件 |
| OB86_FLT_ID | BYTE | 故障代码：（可能值 B#16#C1，B#16#C2，B#16#C3，B#16#C4，B#16#C5，B#16#C6，B#16#C7，B#16#C8） |

| 变　　量 | 类　　型 | 描　　述 |
| --- | --- | --- |
| OB86_PRIORITY | BYTE | 优先级，可通过 STEP 7 选择（硬件组态） |
| OB86_OB_NUMBR | BYTE | OB 号（86） |
| OB86_RESERVED_1 | BYTE | 备用 |
| OB86_RESERVED_2 | BYTE | 备用 |
| OB86_MDL_ADDR | WORD | 根据故障代码 |
| OB86_RACKS_FLTD | BOOL ARRAY[0..31] | 根据故障代码 |

**例 6**　下面以一个示例演示如何应用 OB86，通过 OB86 可以得到什么信息。

首先，新建一个项目 OB86_eg，在项目中插入一个 S7－300 站，然后插入 CPU 315－2DP，选择 DP 作为主站，然后双击硬件组态中 CPU 所在的行，并打开 CPU315－2DP 的 Interrupts 选项页，如图 8-22 所示，可以看到这个 CPU 支持 OB86。

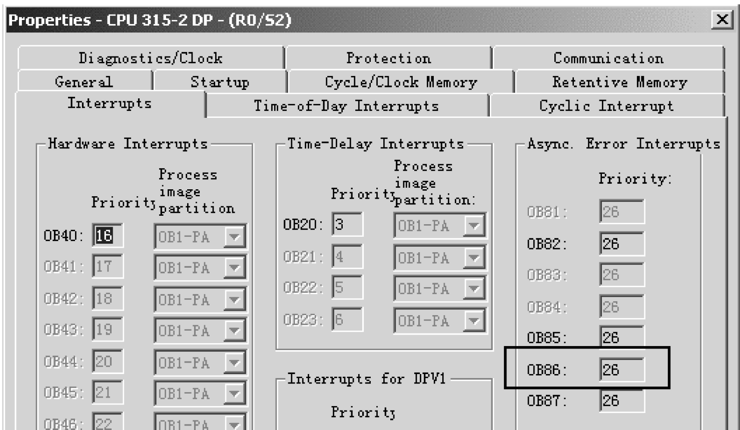



图 8-22　CPU315－2DP 的 Interrupts 选项页

在 DP 主站下面添加一个 ET200M 从站（在硬件目录栏中的 PROFIBUS－DP 下），并在主站中插入一个模拟量模块 SM331（订货号为 6ES7 331－7KF02－0AB0），必须注意 DP 主站地址和 ET200M 从站地址（此处设置为 3，如图 8-23 所示设置 DP 从站属性）不能相同，并且 ET200M 的站地址必须和 ET200M 模块上的实际设置地址一致，硬件组态如图 8-24 所示。

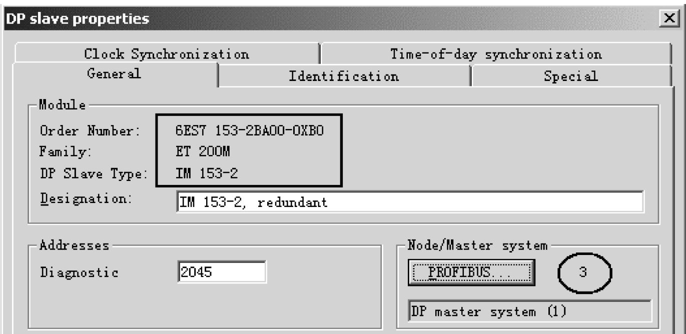



图 8-23　设置 DP 从站属性

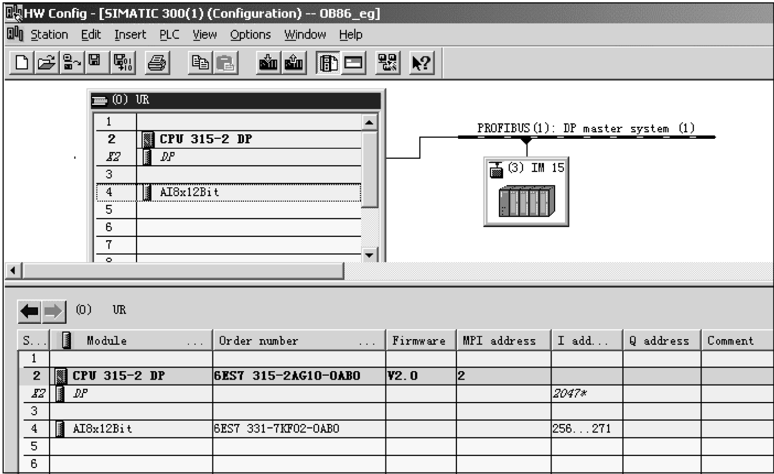



图 8-24　硬件组态

之后打开 OB86，编写机架故障组织块 OB86 程序，如图 8-25 所示。

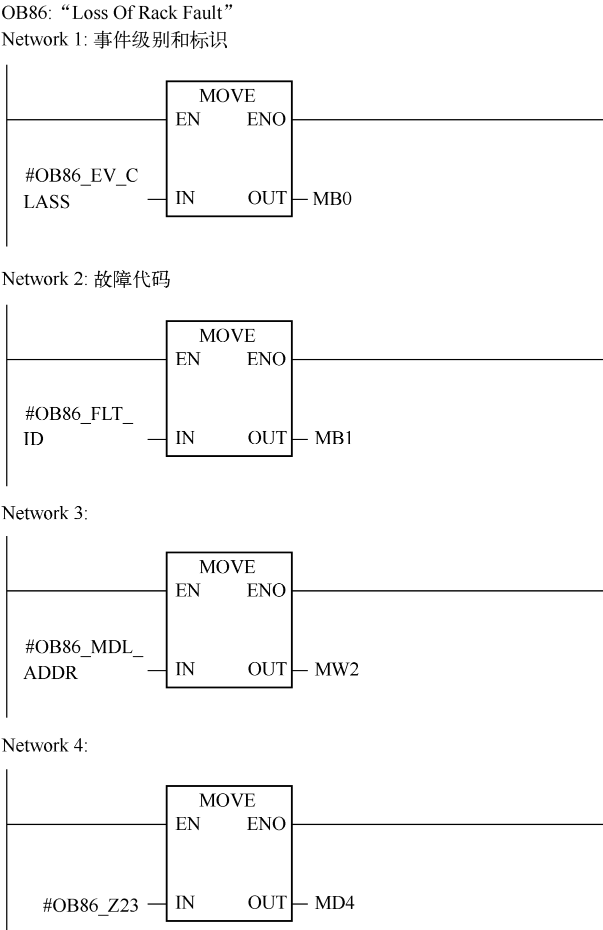



图 8-25　机架故障组织块 OB86 程序

说明：将局部变量 OB86_RACKS_FLTD 改为 OB86_Z23，类型为 DWORD。在程序中通过局部变量 OB86_EV_CLASS、OB86_FLT_ID、OB86_MDL_ADDR 和 OB86_Z23 获得系统的故障信息，分别置于 MB0、MB1、MW2 和 MD4 中。

将 OB86 和硬件组态下载到 CPU 中。插入变量表，填入存储器地址 MB0、MB1、MW2、MD4。设置好故障后，利用变量表监控程序的运行，如图 8-26 所示。此时可以读到 MB0 为"B#16#39"，表示发生故障到来的事件；MB1 为"B#16#C4"，可知 DP 站有故障；MW2 为"W#16#07FFH"，表示主站逻辑地址为 2047；MD4 为"DW#16#07FD0103H"，其中"0～7"位表示出现错误的从站地址为 3，"16～30"位表示从站逻辑地址为 2045（07FDH）。

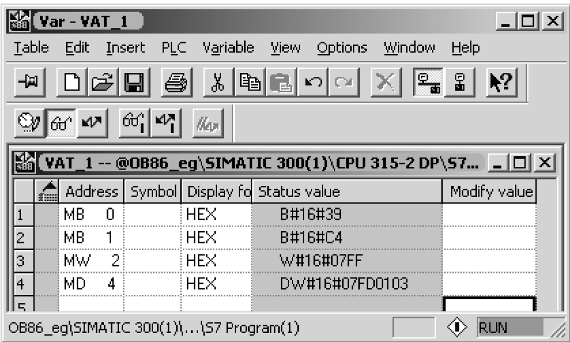



图 8-26　利用变量表监控程序的运行

### 8.7.5　通信错误组织块（OB87）

在使用通信功能块或全局数据（GD）通信进行数据交换时，若出现如下几种错误时，系统将调用 OB87：

① 接收全局数据时，检测到不正确的帧标识符。

② 全局数据通信的状态信息数据块不存在或太短。

③ 接收到非法的全局数据包编号。

如果用于全局数据通信状态信息的数据块丢失，需要用 OB87 生成该数据块并将它下载到 CPU 中。

## 8.8　起动组织块（OB100～OB102）

在 STEP 7 中，支持三种起动程序组织块：OB100（暖起动）、OB101（热起动）和 OB102（冷起动）。根据起动事件及不同的 CPU，相应的起动 OB 块被调用。

对于大多数的 S7-300 CPU 来说，起动类型为暖起动，起动时过程映像和不保持的定时器、计数器及标志存储器被清除，然后程序从 OB1 的第一条指令开始执行。暖起动包括两种方式：自动暖起动方式和手动暖起动方式。前者指 CPU 上电的起动；后者指操作模式改变为运行状态，这是通过 CPU 上的模式选择开关或利用编程设备中的 STEP 7 软件而设置的。在循环程序 OB1 执行之前，要执行起动组织块中的程序，且只执行一次。用户可以利用这个性质，在起动组织块中编写程序，例如可用于变量的初始化，预置通信连接等。

S7－400 还支持热起动组织块 OB 101。在热起动时，所有数据（过程映像、定时器、计数器及标志存储器）被保持，程序从断点处恢复执行。

CPU318－2 和 CPU 417－4 还具有冷起动 OB102 的起动方式。针对电源故障可以定义这种附加的起动方式。它是通过硬件组态时的 CPU 参数而设置的。冷起动时，所有过程映像和定时器、计数器及标志存储器被清除，数据块保持其预置值。首先执行起动组织块 OB102，然后从 OB1 的第一条指令开始执行。

起动组织块的局部变量如表 8-15 所示。

**表 8-15　起动组织块的局部变量**

| 变　　量 | 类　　型 | 描　　述 |
|---|---|---|
| OB10_EV_CLASS | BYTE | 事件级别和标识：B#16#13：激活 |
| OB10x_STRTUP | BYTE | 起动请求：B#16#81：手动暖起动<br>B#16#82：自动暖起动<br>B#16#83：手动热起动请求<br>B#16#84：自动热起动请求<br>B#16#85：手动冷起动请求<br>B#16#86：自动冷起动请求 |
| OB10x_PRIORITY | BYTE | 优先级：27 |
| OB10x_OB_NUMBR | BYTE | OB 号（100，101，或 102） |
| OB10x_STOP | WORD | 引起 CPU 停机事件的号码 |
| OB10x_STRT_INFO | DWORD | 关于当前起动的进一步信息 |
| OB10x_DATE_TIME | DATE_AND_TIME | OB 被调用时的日期和时间 |

注：x 为 0，1，2。

**例 7**　在 S7－300 中只有一个起动组织块 OB100。通过分析 OB100 的起动信息，可以在程序中确定起动的类型。试在 OB100 中编程使得当手动暖起动时输出 Q0.0 被置位，而当自动暖起动时 Q 0.1 被置位。提示：利用起动组织块的局部变量 OB100_STRTUP。

说明：在编程时利用比较指令，比较局部变量 OB100_ STRTUP 中的值是否与起动请求相同。手动暖起动的起动请求为 B#16#81，自动暖起动的起动请求为 B#16#82。起动组织块 OB100 程序如下：

```
Network1:是否为手动暖起动方式
L     #OB100_STRTUP
L     B#16#81
==I
=   Q0.0
Network1:是否为自动暖起动方式
L     #OB100_STRTUP
L     B#16#82
==I
=   Q0.0
```

## 8.9 同步错误组织块

程序中如果有不正确的地址区、错误的编号或错误的地址，都会出现同步错误，操作系统会自动调用同步错误组织块。OB121 用于对程序错误的处理，OB122 用于处理模块访问错误。同步组织块的优先级与检测到出错块的优先级相同，所以它们可以访问中断发生时累加器和其他寄存器中的内容。如表 8-16 所示是调用同步错误组织块的典型案例。

表 8-16 调用同步错误组织块的典型案例

| 错误类型 | 案例 |
| --- | --- |
| 编程错误 OB121 | 在程序中调用一个 CPU 中并不存在的块 |
| 访问错误 OB122 | 访问一个有故障模块或不存在的模块（例如，直接访问一个不存在的 I/O 模块） |

### 8.9.1 编程故障组织块（OB121）

当有关程序处理的故障事件发生时，CPU 的操作系统调用 OB121。OB121 与被中断的块在同一优先级中执行。如果 OB121 未编程，CPU 则从 RUN 方式进入 STOP 方式。如表 8-17 所示描述了 OB121 的局部变量。

表 8-17 OB121 的局部变量

| 变量 | 类型 | 描述 |
| --- | --- | --- |
| OB121_EV_CLASS | BYTE | 事件级别和标识：B#16#25 |
| OB121_SW_FLT | BYTE | 故障代码：（可能值：B#16#21，B#16#22，B#16#23，B#16#24，B#16#25，B#16#26，B#16#27，B#16#28，B#16#29，B#16#30，B#16#31，B#16#32，B#16#33，B#16#34，B#16#35，B#16#3A，B#16#3C，B#16#3D，B#16#3E，B#16#3F） |
| OB121_PRIORITY | BYTE | 优先级 = 出现故障的组织块的优先级 |
| OB121_BLK_TYPE | BYTE | 出现故障的块的类型（在 S7-300 时无有效值在这里记录）：B#16#88：OB，B#16#8A：DB，B#16#8C：FB，B#16#8E：FB |

**例 8** 当 CPU 调用一个未下载到 CPU 的程序块时，CPU 会调用 OB121，通过局部变量 OB121_BLK_TYPE 可以得出出现错误的程序块。通过这个案例，使用户熟悉 OB121 的用法。

建立新项目 OB121_eg，在“Blocks”中插入 OB121 和 FC1，在其中分别编写程序。

说明：如图 8-27 所示为编程故障组织块 OB121 程序，将局部变量 OB121_BLK_TYPE 的值存入 MW0；如图 8-28 为 FC1 程序，而且在 FC1 中建立两个局部变量 in 和 out，in 控制 out 的通断；如图 8-29 为 OB1 程序，有条件（M10.0）调用 FC1，当 M10.0 为 1 时，通过 M20.1 控制 M20.2。

先将硬件组态和 OB1 下载到 CPU 中，此时 CPU 能正常运行。在程序的 Blocks 中插入 Variable Table，填入存储器地址 MW0 和 M10.0，并点击工具栏中按钮，程序运行正常，若将 M10.0 置为 true，则 CPU 报错并停机。查看 CPU 的诊断缓冲区信息，如图 8-30 所示，发现为编程错误。将 OB121 下载到 CPU 中。再将 M10.0 置为 true，CPU 会报错误但不会停机，此时 MW0 显示为 B#16#88，查看手册得知故障代码 B#16#88 表示为 OB 程序错误。检

查发现 FC1 未下载，而且当 M20.1 通时，M20.2 仍保持不得电。下载 FC1 后再将 M10.0 置为 true，CPU 不会再报错，程序也不会再调用 OB121。此时当 M20.1 通时，M20.2 将得电。

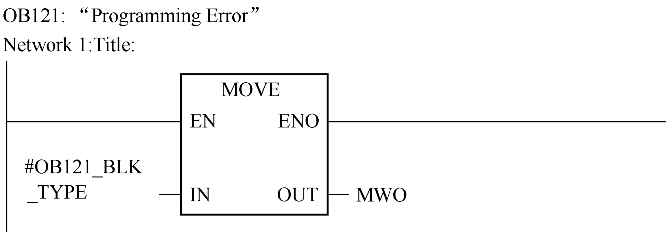



图 8-27　编程故障组织块 OB121 程序




图 8-28　FC1 程序

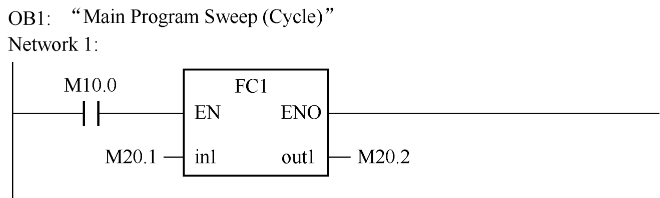



图 8-29　OB1 程序

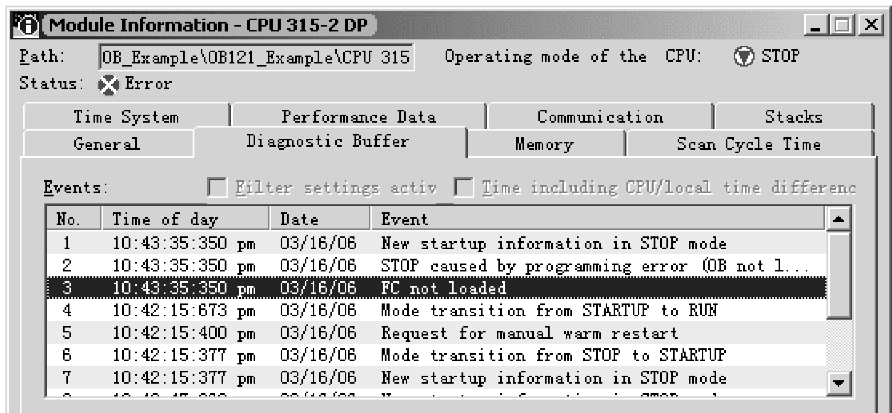



图 8-30　CPU 诊断缓冲区信息

## 8.9.2　I/O 访问故障组织块（OB122）

当对模板的数据访问出现故障时 CPU 的操作系统调用 I/O 访问故障组织块 OB122。例如，直接访问 I/O 错误（模块损坏或找不到），或者访问了一个 CPU 不能识别的 I/O 地址，

此时操作系统将会调用 OB122。OB122 与被中断的块在同一优先级中执行。如果 OB122 未编程，CPU 从 RUN 方式进入 STOP 方式。

当出现模块访问错误时，将出错模块的地址记录在一个内存字中。之后使 CPU 进入停机（STOP）状态。此时可以从与其相连的 OP（操作面板）上读出这个内存字的内容。OB122 的局部变量如表 8-18 所示。

表 8-18　OB122 的局部变量

| 变　　量 | 类　　型 | 描　　述 |
| --- | --- | --- |
| OB122_EV_CLASS | BYTE | 事件类别，IDs = B#16#29 |
| OB122_SW_FLT | BYTE | 错误代码：（可能的值为：B#16#42，B#16#43，B#16#44，B#16#45） |
| OB122_PRIORITY | BYTE | 发生错误的组织块的优先级 |
| OB122_OB_NUMBR | BYTE | OB 块的号码（122） |
| OB122_BLK_TYPE | BYTE | 发生错误的块的类型：B#16#88：OB？B#16#8A：DB？B#16#8C：FC？B#16#8E：FB？ |
| OB122_MEM_ADDR | WORD | 发生错误处的内存地址 |
| OB122_BLK_NUM | WORD | 导致错误的 MC7 指令所在块的地址 |
| OB122_PRG_ADDR | WORD | 导致错误的 MC7 指令的相对地址 OB122_ DATE_ TIME |

## 8.10　系统功能

系统功能简称为 SFC，是系统为用户提供的具有一定功能的程序块。SFC 的功能一览表如表 8-19 所示。SFC 集成在 S7 CPU 的操作系统中，属于操作系统的一部分，不占用程序空间。这些程序块已经编译好，就像 C 语言中的系统子程序一样，系统只公布其功能和输入/输出参数，用户看不到其程序代码。根据其功能，用户在自己的程序中可以随意调用，但不能做任何改动。

表 8-19　SFC 的功能一览表

| 功　　能 | SFC 编号 | SFC 名称 | 意　　义 |
| --- | --- | --- | --- |
| 拷贝与块功能 | SFC20 | BLKMOV | 拷贝变量 |
| | SFC81 | UBLKMOV | 不间断地复制变量 |
| | SFC21 | FILL | 初始化存储区 |
| | SFC22 | CREAT_DB | 生成数据块 |
| | SFC23 | DEL_DB | 删除数据块 |
| | SFC24 | TEST_DB | 测试数据块 |
| | SFC25 | COMPRESS | 压缩用户存储器 |
| | SFC44 | REPL_VAL | 传送一个替代值到累加器 1 |
| | SFC82 | CREA_DBL | 在装载存储器中生成数据块 |
| | SFC83 | READ_DBL | 从装载存储器的数据块中读取数据 |
| | SFC84 | WRIT_DBL | 写数据到装载存储器中的数据块 |

（续）

| 功　　能 | SFC 编号 | SFC 名称 | 意　　义 |
| --- | --- | --- | --- |
| 用于控制程序执行 | SFC43 | RE_TRIGR | 重新触发循环时间监控 |
| | SFC46 | STP | 使 CPU 进入停机状态 |
| | SFC47 | WAIT | 延迟用户程序执行 |
| | SFC35 | MP_ALM | 触发多处理器中断 |
| | SFC104 | CiR | 控制 CiR |
| 用于控制系统时钟 | SFC0 | SET_CLK | 设定 TOD |
| | SFC1 | READ_CLK | 读取时间 |
| | SFC48 | SNC_RTCB | 同步子时钟 |
| | SFC 100 | SET_CLKS | 设定日期时间和时间日期状态 |
| 用于控制运行时间定时器 | SFC101 | RTM | 控制运行时间定时器 |
| | SFC2 | SET_RTM | 设定运行时间定时器 |
| | SFC3 | CTRL_RTM | 启/停运行时间定时器 |
| | SFC4 | READ_RTM | 读取运行时间定时器 |
| | SFC64 | TIME_TCK | 读取系统时间 |
| 用于传输数据记录 | SFC54 | RD_DPARM | 读取定义的参数 |
| | SFC102 | RD_DPARA | 读取预定义的参数 |
| | SFC55 | WR_PARM | 写动态数据 |
| | SFC56 | WR_DPARM | 写缺省参数 |
| | SFC57 | PARM_MOD | 分配模块参数 |
| | SFC58 | WR_REC | 写数据记录 |
| | SFC59 | RD_REC | 读数据记录 |
| 用于日期时间中断 | SFC28 | SET_TINT | 设置日期时间中断 |
| | SFC29 | CAN_TINT | 取消日期时间中断 |
| | SFC30 | ACT_TINT | 起动日期时间中断 |
| | SFC31 | QRY_TINT | 查询日期时间中断 |
| 处理延时中断 | SFC32 | SRT_DINT | 起动延时诊断 |
| | SFC34 | QRY_DINT | 查询一个延时诊断 |
| | SFC33 | CAN_DINT | 取消一个延时诊断 |
| 处理同步故障 | SFC36 | MSK_FLT | 屏蔽同步故障 |
| | SFC37 | DMSK _FLT | 解除同步故障的屏蔽 |
| | SFC38 | READ_ERR | 读取故障寄存器中的信息 |
| 处理中断和异步故障 | SFC39 | DIS_IRT | 禁止新的中断和异步故障的处理 |
| | SFC40 | EN_IRT | 激活新的中断和异步故障的处理 |
| | SFC41 | DIS_AIRT | 延迟一个高优先权的中断和异步故障的处理 |
| | SFC42 | EN_AIRT | 激活具有高优先权的中断和异步故障的处理 |

（续）

| 功　　能 | SFC 编号 | SFC 名称 | 意　　义 |
|---|---|---|---|
| 用于诊断 | SFC6 | DR_SINFO | 读取 OB 起动信息 |
| | SFC51 | RDSYSST | 读取系统状态信息表或部分状态信息表 |
| | SFC52 | WR_USMSG | 在诊断缓冲器中写入一个用户定义的诊断事件 |
| | SFC78 | OB_RT | 确定 OB 程序运行时间 |
| | SFC87 | C_DIAG | 诊断当前的连接状态 |
| | SFC103 | DP_TOPOL | 识别 DP 主站系统的总线拓扑结构 |
| 用于刷新过程映像和处理位区域 | SFC26 | UPDAT_PI | 刷新过程映像输入表 |
| | SFC27 | UPDATE_PO | 刷新过程映像输出表 |
| | SFC126 | SYNC_PI | 同步刷新过程映像区输入表 |
| | SFC127 | SYNC _PO | 同步刷新过程映像区输出表 |
| | SFC79 | SET | 置位 I/O 区域中的位区域 |
| | SFC80 | RSET | 复位 I/O 区域中的位区域 |
| 分布式 I/O | SFC7 | DP_PRAL | 在 DP 主站上触发硬件中断 |
| | SFC11 | DPSYC_FR | 同步 DP 从站组 |
| | SFC12 | D_ACT_D | 取消和激活 DP 从站 |
| | SFC13 | DPNRM_DG | 读 DP 从站诊断数据(从站诊断) |
| | SFC14 | DPRD_DAT | 读取 DP 标准从站的连续数据 |
| | SFC15 | DPWR_DAT | 向 DP 标准从站写连续数据 |
| 用于全局数据通信 | SFC60 | GD_SND | 传送一个全局数据包 |
| | SFC61 | GD_RCV | 接收全局数据包 |
| 用于模板寻址 | SFC5 | GADR_LGC | 查询模板的逻辑起始地址 |
| | SFC49 | LGC_GADR | 查询逻辑地址所属的插槽 |
| | SFC50 | RD_LGADR | 查询一个模板所有的逻辑地址 |
| 用于非组态 S7 连接通信 | SFC65 | X_SEND | 发送数据到不属于本地 S7 站的通信对象 |
| | SFC66 | X_RCV | 接收不属于本地 S7 站的通信对象的数据 |
| | SFC68 | X_PUT | 写数据到不属于本地 S7 站的通信对象 |
| | SFC67 | X_GET | 读不属于本地 S7 站的通信对象的数据 |
| | SFC69 | X_ABORT | 中断一个不属于本地 S7 站已建立的连接 |
| | SFC73 | I_PUT | 写数据到本地 S7 站的通信对象 |
| | SFC72 | I_GET | 读本地 S7 站的通信对象的数据 |
| | SFC74 | I_ABORT | 中断一个与本地 S7 站已建立的连接 |
| 生成块相关的信息 | SFC10 | DIS_MSG | 禁止块相关的、符号相关的以及组状态信息 |
| | SFC9 | EN_MSG | 使能块相关的、符号相关的以及组状态信息 |
| | SFC17 | ALARM_SQ | 生成可确认的与块相关的信息和用 SFC18 “ALARM_S”生成永久确认的块相关的信息 |
| | SFC19 | ALARM_SC | 查询最后 ALARM_SQ/ALARM_DQ 生成可确认的与永久确认的块相关的信息 |
| | SFC107<br>SFC108 | ALARM_DQALARM_D | 生成可确认的与永久确认的块相关的信息 |
| | SFC105 | READ_SI | 读取动态系统资源 |
| | SFC106 | DEL_SI | 删除动态系统资源 |

系统功能块简称为 SFB，也是系统为用户提供的具有一定功能的程序块。SFB 的功能一览表如表 8-20 所示。与 SFC 相比，SFB 有存储功能，其变量保存在指定给它的背景数据块中。

**表 8-20　SFB 的功能一览表**

| 功　　能 | SFB 编号 | SFB 名称 | 意　　义 |
| --- | --- | --- | --- |
| 符合 PNO AK1131 的 DPV1 | SFB52 | RDREC | 读来自 DP 从站的数据记录 |
| | SFB53 | WRREC | 向 DP 从站写数据记录 |
| | SFB54 | RALRM | 接收来自 DP 从站的中断 |
| | SFB75 | SALRM | 向 DP 从站发送中断 |
| S7 通信 | SFB8 | U_SEND | 非协调发送数据 |
| | SFB9 | U_RECV | 非协调接收数据 |
| | SFB12 | B_SEND | 发送段数据 |
| | SFB13 | B_RCV | 接收段数据 |
| | SFB15 | PUT | 写数据到远程 CPU |
| | SFB14 | GET | 读远程 CPU 数据 |
| | SFB16 | PRINT | 发送数据到打印机 |
| | SFB19 | START | 在远程设备上初始化一个暖或冷起动 |
| | SFB20 | STOP | 停止远程设备 |
| | SFB21 | RESUME | 在远程设备上初始化一个热起动 |
| | SFB22 | STATUS | 查询远程对象的状态 |
| | SFB23 | USTATUS | 接收远程设备的状态 |
| | SFB62 | CONTROL | 查询连接的状态 |
| 生成块相关的信息 | SFB36 | NOTIFY | 生成无需确认的块相关的信息 |
| | SFB31 | NOTIFY_8P | 生成无需显示确认的块相关的信息 |
| | SFB33 | ALARM | 生成需确认的块相关的信息 |
| | SFB35 | ALARM_8P | 生成 8 个信号的带相关数据的块相关的信息 |
| | SFB34 | ALARM_8 | 生成 8 个信号的不带相关数据的块相关的信息 |
| | SFB37 | AR_SEND | 发送存档数据 |
| IEC 定时器和 IEC 计数器 | SFB3 | TP | 生成一个脉冲信号 |
| | SFB4 | TON | 生成一个延时接通信号 |
| | SFB5 | TOF | 生成一个延时断开信号 |
| | SFB0 | CTU | 实现加计数功能 |
| | SFB1 | CTD | 实现减计数功能 |
| | SFB2 | CTUD | 实现加/减计数功能 |
| 用于集成控制功能 | SFB41 | CONT_C | 实现连续调节功能 |
| | SFB42 | CONT_S | 实现步进调节功能 |
| | SFB43 | PULSEGEN | 实现脉冲发生功能 |

（续）

| 功　　能 | SFB 编号 | SFB 名称 | 意　　义 |
| --- | --- | --- | --- |
| 用于紧凑型 CPU | SFB44 | ANALOG | 实现模拟量输出定位 |
| | SFB46 | DIGITAL | 实现数字量输出定位 |
| | SFB47 | COUNT | 控制计数器 |
| | SFB48 | FREQUENC | 控制频率测量 |
| | SFB49 | PULSE | 控制脉宽调制 |
| | SFB60 | SEND_PTP | 发送数据（ASCII，3964（R）） |
| | SFB61 | RCV_PTP | 接收数据（ASCII，3964（R）） |
| | SFB62 | RES_RCVB | 清除接收缓冲区（ASCII，3964（R）） |
| | SFB63 | SEND_RK | 发送数据（512（R）） |
| | SFB64 | FETCH_RK | 获取数据（RK 512） |
| | SFB65 | SERVE_RK | 接收和提供数据（RK 512） |
| 集成功能（用于集成 I/O 的 CPU） | SFB29 | HS_COUNT | 对计数器功能产生影响 |
| | SFB30 | FREQ_MES | 对频率测量计功能产生影响 |
| | SFB38 | HSC_A_B | 对 A/B 计数器功能产生影响 |
| | SFB39 | POS | 对定位功能产生影响 |
| 汇编技术 | SFB63 | AB_CALL | 调用一个汇编代码块 |
| 用于刷新过程映像和处理位区域 | SFB32 | DRUM | 执行一段顺序程序 |

SFC 和 SFB 的具体功能详见光盘中所附系统手册《SIMATIC S7－300/400 的系统软件和标准功能》。在介绍 OB 块的使用时，我们已经使用了一些相关的 SFC，下面再介绍几个结合网络功能使用的 SFC。

**例 9**　将 S7－300 模块上的输入数据 IB0 发送到 S7－200 模块，并通过输出 QB0 显示；S7－300 将 S7－200 模块上的输入数据 IB0 读回到本地并通过输出 QB0 显示。要求采用无组态连接方式中的单边编程通信方式的实现。

通过 MPI 实现 PLC 到 PLC 之间的通信有三种方式：全局数据包通信方式、无组态连接方式和组态连接方式。

全局数据包通信方式是在配置 PLC 硬件组态的过程中，设置通信的 PLC 站之间的发送区和接收区，无需任何的程序处理，这种方式只适合 S7－300/400 PLC 之间的通信。

无组态连接方式需要调用系统功能块 SFC65～SFC69 来实现，这种通信方式适合于 S7－300、S7－400 和 S7－200 之间的通信。通过调用 SFC 来实现的 MPI 通信又可分为两种方式：双边编程通信方式和单边编程通信方式。双边通信方式指在通信的双方都需要调用通信块，一方调用发送块 SFC65 发送数据，另一方通过调用接收块 SFC66 接收数据，这种方式只适合 S7－300/400 PLC 之间的通信。单边编程通信方式只在一方编写通信程序，即客户（需编写程序的 CPU）与服务器（无需编写程序的 CPU）的访问模式，这种通信方式适合 S7－300/400/200 PLC 之间的通信，而且 S7－200 只能作为服务器，编写程序时需使用系统功能块 SFC67 和 SFC68。

组态连接方式只适合 S7 -300/400 PLC 以及 S7 -400/400 PLC 之间的通信，需要调用 SFB14 和 SFB15 来实现。S7 -300/400 通信时，S7 -300 只能作为服务器，S7 -400 作为客户机对 S7 -300 的数据进行读写操作；S7 -400/400 通信时，一个 S7 -400 作为服务器，另一个作为客户机。

本例应用无组态编程通信方式，所以仅需对 S7 -200 进行设置而无需编程；在 S7 -300 中需调用 SFC68 “X_PUT”（参数说明见表 8-21）和 SFC67 “X_GET”（参数说明见表 8-22）进行通信。SFC68 的功能是写数据到不属于本地 S7 站的通信对象，SFC67 的功能是读不属于本地 S7 站的通信对象的数据。

**表 8-21　SFC68 的参数说明**

| 参　　数 | 声　　明 | 数据类型 | 存储区域 | 描　　述 |
|---|---|---|---|---|
| REQ | INPUT | BOOL | I，Q，M，D，L，常数 | 控制参数“请求激活” |
| CONT | INPUT | BOOL | I，Q，M，D，L，常数 | 控制参数“继续” |
| DEST_ID | INPUT | WORD | I，Q，M，D，L，常数 | 地址参数“目标 ID”，包含了对方的 MPI 地址 |
| VAR_ ADDR | INPUT | ANY | I，Q，M，D | 发送到对方 CPU 的数据地址区域。必须选择通信对象支持的数据类型 |
| SD | INPUT | ANY | I，Q，M，D | 发送数据。允许下面的类型：BOOL，BYTE，CHAR，WORD，INT，DWORD，DINT，REAL，DATE，TOD，TIME，S5_TIME，DATE_AND_TIME 以及除了 BOOL 外上述数据类型的数组<br>SD 必须和 VAR_ ADDR 相同长度，数据类型也必须匹配。最大发送数据长度为 84 字节 |
| RET_VAL | OUTPUT | INT | I，Q，M，D，L | 功能调用时如故障发生，返回值包含故障代码 |
| BUSY | OUTPUT | BOOL | I，Q，M，D，L | 为 1，表示发送未完成；为 0，表示发送完成或无发送功能激活 |

**表 8-22　SFC67 的参数说明**

| 参　　数 | 声　　明 | 数据类型 | 存储区域 | 描　　述 |
|---|---|---|---|---|
| REQ | INPUT | BOOL | I，Q，M，D，L，常数 | 控制参数“请求激活” |
| CONT | INPUT | BOOL | I，Q，M，D，L，常数 | 控制参数“继续” |
| DEST_ID | INPUT | WORD | I，Q，M，D，L，常数 | 地址参数“目标 ID”，包含了对方的 MPI 地址 |
| VAR_ADDR | INPUT | ANY | I，Q，M，D | 发送到对方 CPU 的数据地址区域。必须选择通信对象支持的数据类型 |
| RET_VAL | OUTPUT | INT | I，Q，M，D，L | 功能调用时如故障发生，返回值包含故障代码。如没有故障代码，RET_VAL 将返回所接收的数据长度（以字节为单位） |
| BUSY | OUTPUT | BOOL | I，Q，M，D，L | 为 1，表示发送未完成；为 0，表示发送完成或无发送功能激活 |
| RD | OUTPUT | ANY | I，Q，M，D | 发送数据。允许下面的类型：BOOL，BYTE，CHAR，WORD，INT，DWORD，DINT，REAL，DATE，TOD，TIME，S5_TIME，DATE_AND_TIME 以及除了 BOOL 外上述数据类型的数组。RD 接收区的长度必须与通信对象的 VAR_ADDR 的一致，数据类型也必须匹配。最大接收区域长度为 96B |

（1）硬件需求

CPU313C－2DP，CPU226，插入 PROFIBUS 网卡 CP5611 的 PC 和带三个 DP 接头的 MPI 电缆（即 PROFIBUS 电缆）。

（2）物理连接

将 MPI 电缆两端接头上的拨码拨至 ON，中间接头的拨码拨至 OFF。利用这根电缆将 PC、CPU 313C－2DP 和 CPU 226 建立 MPI 网络物理连接图。

（3）网络的硬件组态

1）组态 CPU 313C－2DP：在 STEP7 中命名为“mpi_ 300”。在此项目中新建一个站，命名为 STATION 300。将其 MPI 站地址设定为 2，通信速率设定为 187.5kbit/s，保存并下载。

2）组态 CPU 226：在 MicroWin 4.0 中，对 CPU226 进行参数设置，定站地址为 4，通信速率为 187.5 kbit/s，设置完毕后下载。注意：下载 CPU200 的 PG/PC 设置如图 8－31 所示。

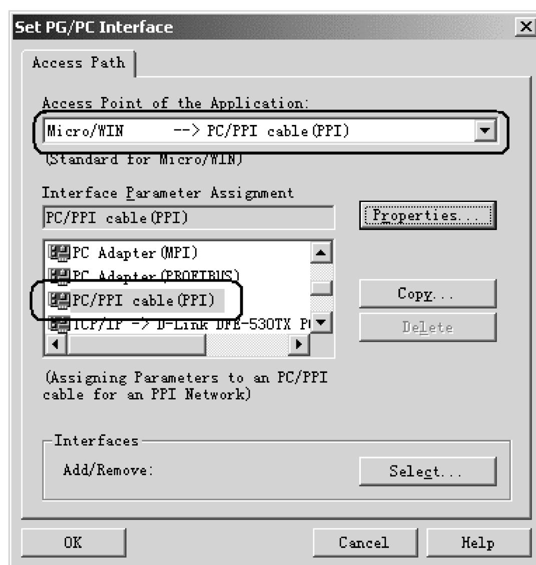



图 8-31　CPU200 的下载设置

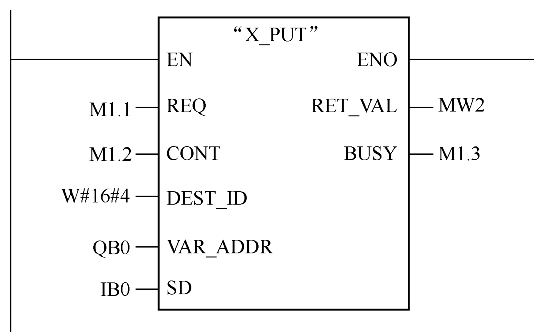
（4）软件编程与下载

在 STEP 7 软件中为 CPU 313－2DP 编写程序，在 OB1 中调用 SFC67 和 SFC68，MPI 网络通信的程序如图 8-32 所示。

说明：Network1 中，当 M1.1 为 1 时，将 S7－300 中的输入数据 IB0 发送到 S7－200 的 QB0 中，VAR_ ADDR 表示发送到对方 CPU 的数据地址区域（本程序指 CPU226 的接收数据区域）；Network2 中，当 M1.4 为 1 时，S7－300 将 S7－200 的输入数据 IB0 读回到本地输出数据 QB0 中。

在 SIMATIC 管理器窗口中，选中“Blocks”中的每一个程序块，点击工具栏中图标，完成软件下载。

Network 1: 发送数据




Network 2: 接收数据

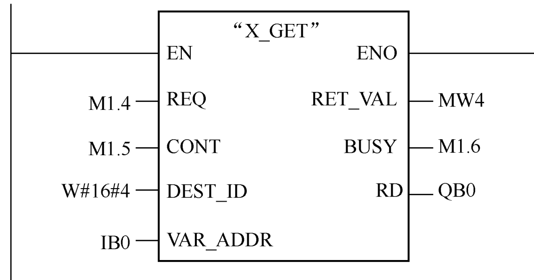



图 8-32　S7 -300 中的 OB1 程序

（5）程序的监控

建立监控变量表 VAT1，插入 4 个 CPU 313C -2DP 的变量 IB0、M1. 1、QB0 和 M1. 4，利用变量表监控程序的运行，如图 8-33 所示。首先将 IB0 赋初值 B#16#0B，然后变量表中的发送请求变量 M1. 1 强制为 1，即可监控到 CPU 226 外部硬件上接收到了 CPU 313C -2DP 的发送数据 B#16#0B，并通过 CPU 226 的 QB0 输出。然后将接收请求变量 M1. 4 强制为 1，同时设置 CPU 226 外部硬件的输入端，即可监控到 CPU 313C -2DP 读取到了 CPU 226 中的数据 B#16#03，并通过 CPU 313C -2DP 的 QB0 输出。

Var - VAT_1

Table Edit Insert PLC Variable View Options Window Help

VAT_1 -- @mpi_300\SIMATIC 300(1)\CPU 313C-2 DP\S7 Program(1) ONLINE

| | Address | Symbol | Display format | Status value | Modify value |
|---|---|---|---|---|---|
| 1 | IB 0 | | HEX | B#16#0B | B#16#0B |
| 2 | M 1.1 | | BOOL | true | true |
| 3 | QB 0 | | HEX | B#16#03 | |
| 4 | M 1.4 | | BOOL | true | true |
| 5 | | | | | |

图 8-33　利用变量表监控程序的运行

**例 10**　在两个 S7 -300 模块中的 DB 区的数据进行通信，采用 PROFIBUS 打包通信方式。

PROFIBUS 网络组态的方法见第 9 章，其中将图 9-25 中的“长度”设定为 10。注意：设置好主站和从站的发送通信区和接收通信区，本例中主站输出区 QB0 ~ QB9 对应从站输入区 IB0 ~ IB9；从站输出区 QB5 ~ QB14 对应主站输入区 IB5 ~ IB14。

编写主站和从站程序时，要求分别建立 OB1、OB82、OB86、OB122，其中 OB82、OB86、OB122 是为了避免网络某个站点掉电而使整个网络不能正常工作。如图 8-34 所示为主站程序 OB1，如图 8-35 所示为从站程序 OB1。

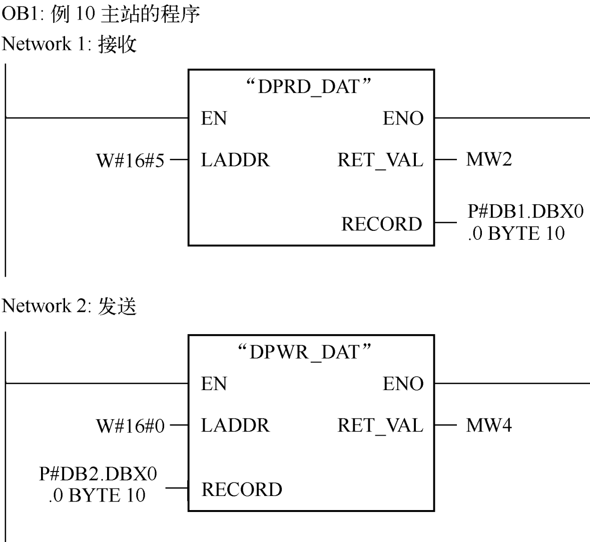



图 8-34　主站程序 OB1

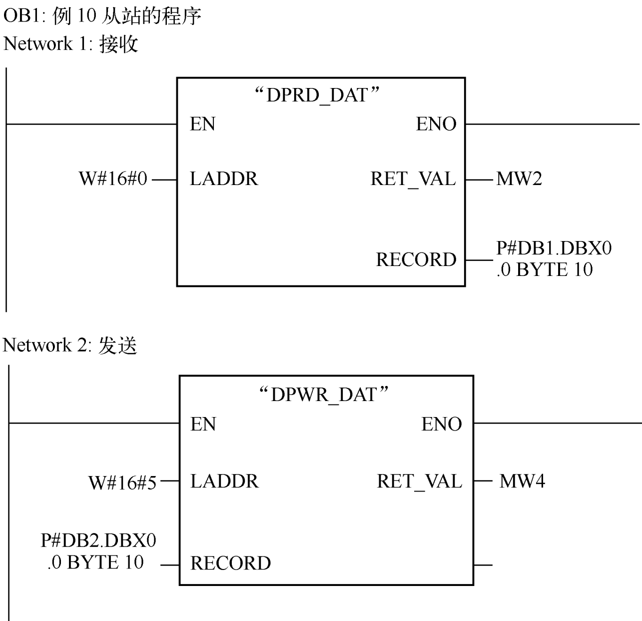



图 8-35　从站程序 OB1

说明：在 S7－300 主站和从站中都调用 SFC14“DPRD_DAT”解包并接收数据（参数说明见表 8-23）和 SFC15“DPWR_DAT”打包并发送数据（参数说明见表 8-24），实现主从站之间的通信。在主站程序 Network1 中，SFC14 的 RECORD 端设置 ANY 指针格式的数据为 P#DB1. DBX0. 0 BYTE 10，LADDR 端设置为 W#16#5，表示主站解开存放在 IB5～IB14 的数据包，并存放在 DB1. DBB0～DB1. DBB9 中；Network2 中，表示 SFC15 给存放在主站 DB2. DBB0～DB1. DBB9 的数据打包，通过 QB0～QB9 发送出去。从站的程序与主站程序相似，用户可以自己分析其功能。

**表 8-23　SFC14“DPRD_DAT”的参数说明**

| 参数 | 声明 | 数据类型 | 存储区域 | 描述 |
|---|---|---|---|---|
| LADDR | INPUT | WORD | I，Q，M，D，L，常数 | 模板输入区中将被读取的数据的起始地址注意：地址必须以十六进制格式输入。例如：诊断地址 100 意味着 LADDR：＝16#16#64 |
| RET_VAL | OUTPUT | INT | I，Q，M，D，L | 如功能执行有故障发生，返回值包含故障代码 |
| RECORD | OUTPUT | ANY | I，Q，M，D | 读取数据存放的目的区。其长度必须与 STEP 7 中对所选模板的配置一致。只能使用 BYTE 数据类型 |

**表 8-24　SFC15“DPRR_DAT”的参数说明**

| 参数 | 声明 | 数据类型 | 存储区域 | 描　　述 |
|---|---|---|---|---|
| LADDR | INPUT | WORD | I，Q，M，D，L，常数 | 模板输出区中将被写入数据的起始地址注意：地址必须以十六进制格式输入。例如：诊断地址 100 意味着 LADDR：＝16#16#64 |
| RECORD | INPUT | ANY | I，Q，M，D | 待写数据存放的源数据区。其长度必须与 STEP 7 中对所选模板的配置一致。只能使用 BYTE 数据类型 |
| RET_VAL | OUTPUT | INT | I，Q，M，D，L | 如功能执行有故障发生，返回值包含故障代码 |

调试时，在主站和从站中分别建立数据块 DB1 的格式，如图 8－36 所示，同样建立 DB2。

DB1 -- S7_Pro1\SIMATIC 300 Station\CPU313C-2DP(1)

| Address | Name | Type | Initial value |
|---|---|---|---|
| 0.0 | | STRUCT | |
| +0.0 | DB_0 | BYTE | B#16#0 |
| +1.0 | DB_1 | BYTE | B#16#0 |
| +2.0 | DB_2 | BYTE | B#16#0 |
| +3.0 | DB_3 | BYTE | B#16#0 |
| +4.0 | DB_4 | BYTE | B#16#0 |
| +5.0 | DB_5 | BYTE | B#16#0 |
| +6.0 | DB_6 | BYTE | B#16#0 |
| +7.0 | DB_7 | BYTE | B#16#0 |
| +8.0 | DB_8 | BYTE | B#16#0 |
| +9.0 | DB_9 | BYTE | B#16#0 |
| | | | |
| =10.0 | | END_STRUCT | |

图 8-36　建立 DB1 数据块的格式

注意：网络组态时，“consistency”项中（见图 3-23），必须选择为“All”，表示 PROFIBUS 通信使用打包方式，程序需用 SFC14、SFC15 对数据进行打包和解包；若选择“Unit”，表示 PROFIBUS 通信使用直接传输方式，可以直接读输入、输出区。

## 习题

1. 简述异步组织块的作用。
2. 利用循环组织块 OB35 建立振荡电路，周期为 2 s。

# 第 9 章　工业网络通信

## 9.1　概述

目前，许多企业正面临着从分立仪器控制向网络化控制的转变。随着计算机、PLC、通信等技术的飞速发展，基于现场总线、工业以太网等技术的工业网络正逐渐被应用到工业现场。现代企业要求建立一个从现场级到企业级的网络结构，通过该网络，现场级的数据可以利用抗干扰性能强的现场总线实现实时的交换、报警、归档、打印等，工作人员可以通过工业以太网对远程控制点的温度、压力、流量等参数实现车间级的远程控制，并能够随时监控现场设备的执行情况，对远程控制点的数据进行归档整理、保存，随时准备调用。而在企业级层面上，各个部门通过普通的以太网实现信息共享、协调工作；甚至可以将现场的数据通过互联网实现异地传输与监控。工业以太网和现场总线技术是工业网络中的两大关键技术。这两者的性能好坏直接影响了整个企业网络的稳定性、快速性等指标。

现场总线技术是随着电子、仪器仪表、计算机技术和网络技术的发展而于 20 世纪 80 年代中期产生的，现场总线技术以其鲜明的特点和优点很快进入各个领域。国外各大控制设备制造商也相继开发了不同的现场总线，但这些现场总线难以形成统一的标准，这从一定程度上影响了现场总线的推广应用。

目前，没有一家现场总线设备制造商能提供所有现场总线产品。国内有少量的石化和化工装置采用了现场总线控制系统（FCS）；FCS 完全不同于 DCS 和 PLC。DCS 和 PLC 是以 I/O 和处理器为核心的，而 FCS 是以智能化的现场仪表和设备为核心的；DCS 和 PLC 的控制任务完全由处理器来完成，而 FCS 的控制任务基本上由现场仪表和设备来完成；这样既丰富了 DCS 的功能，又推动了现场总线和 FCS 的发展。现场总线和 DCS 的集成有两种方式：现场总线与 DCS 的 I/O 总线集成；现场总线与 DCS 网络的集成。前者是现场仪表或现场设备通过现场总线与现场总线接口单元通信，现场总线接口单元再通过 I/O 总线与 DCS 处理器通信。后者是现场仪表或现场设备通过现场总线与现场总线服务器通信，现场总线服务器直接挂在 DCS 网络上。

如今，PLC 在许多设备的控制系统中已得到广泛应用，现场总线的应用也要借助于 PLC。现场总线中，ControlNet、ProfiBus 等本身就是 PLC 的主要供货商支持的，而且这些现场总线技术和产品已集成到 PLC 系统中，成为 PLC 系统中的一部分或者成为 PLC 系统的延伸部分。以 ControlNet 为例，ControlNet 最早由美国 Rockwell Automation 于 1995 年 10 月提出，它是一种高速、高确定性和可重复性的网络，特别适用于对时间有苛刻要求的复杂应用场合的信息传输。

工业以太网技术近年来得到了长足的发展，并已成为事实上的工业标准。随着以太网的通信速率的不断提高和全双工交换式以太网的诞生，以太网的非确定性问题已经解决，使其不仅在企业级甚至车间、现场级也得到了广泛应用。比如 Siemens 的 ProfiNet，它符合

IEEE802 标准，并且采用的是 RJ45 接口屏蔽双绞线传输，现在已经发展成为现场总线的一种，通过 Siemens 的转换设备，可以将原有的现场总线如 PROFIBUS 无缝连接到 ProfiNet 网络中，大大降低企业的网络升级成本，使得工业以太网在工业现场也能得到有效的利用，提高了整个网络的稳定性和快速性。工业以太网大多采用的是 TCP/IP 协议。因为 TCP/IP 协议是互联网广泛采用的协议，因此工业以太网可以非常方便地连接到互联网上。如今许多 DCS 和 PLC 生产厂家纷纷在自己的产品上集成符合 TCP/IP 的工业以太网接口或者推出连接工业以太网的专用模块。

Siemens 公司作为全球知名 PLC 厂商，正在不断改进和提高其 PLC 产品的通信能力和组网能力；为此，Siemens 公司推出了大量的具有不同功能的通信模块，甚至某些 PLC 自身即集成了许多通信接口，用户通过简单的组态，即可组建基于 PROFIBUS 的现场总线网络或者 ProfiNet 的工业以太网络。

目前，Siemens PLC 及其外围扩展设备主要支持 ASI（传感器执行器接口）、MPI、PPI（点对点接口）、自由通信、PROFIBUS、ProfiNet 等常见的通信协议，许多 PLC 本身就同时支持 MPI、PROFIBUS、ProfiNet 协议。以 S7315 -2PN/DP PLC 为例，该 PLC 不仅集成了 ProfiNet 接口还集成了 PROFIBUS-DP、MPI（多点接口）；利用 S7 315 -2PN/DP 既可以组建一个工业以太网，还可以组建 PROFIBUS-DP 的现场总线网。

随着工艺水平和控制要求的不断提高，自动化控制系统已经不仅取决于系统的 CPU，还受到外部网络环境的影响。近几年来，工业通信技术的发展很快，与自动化控制系统不断结合，大大提升了自动化控制系统的性能。

在西门子工业控制网络中，使用了许多通信技术，下面进行简要介绍。

**1. MPI（Multi_Point Interface，多点接口）协议**

MPI 通信用于小范围、少点数的现场级通信，尤其适合作为一种当通信速率要求不高、通信数据量要求不大时应用的简单经济的通信方式，MPI 是为 S7/M7 和 C7 系统提供的多点接口，多用于对 PLC 进行编程、连接上位机和少量 PLC 之间的近距离通信。MPI 通信的物理传输介质与 PROFIBUS 通信所用的紫色电缆线和网络连接器相同，几个 CPU 将其 MPI 编程接口相连，也可以与上位机中的网卡编程接口（MPI/DP）连接。MPI 网络的通信速率支持19.2 Kbit/s ~ 12 Mbit/s。

**2. PROFIBUS 总线**

PROFIBUS 符合国际标准 IEC61158，是目前国际上通用的现场总线标准之一，是用于控制级的通信网络。PROFIBUS 以其独特的技术特点、严格的认证规范、开放的标准、众多厂商的支持和不断发展的应用行规，成为现场级通信网络的最优解决方案。其网络节点数已突破 1000 万个，在现场总线领域遥遥领先。PROFIBUS 网络拓扑形式有总线型、星形、冗余环型，每个网段可以连接多达 32 个接点，PROFIBUS 的传输速率是 1.5 Mbit/s，在总线上可以连接远程 I/O（ET200M、ET200B 等），还可以连接智能从站（S7 -300、S7 -400）。

PROFIBUS 协议包括三个主要部分：

① PROFIBUS-DP：支持分布式外设的通信协议，主站和从站之间采用轮询的通信方式，支持高速的循环数据通信，主要应用于制造业自动化系统中现场级的通信。

② PROFIBUS-PA：支持过程自动化的通信协议，电源和通信数据通过总线并行传输，主要用于面向过程自动化系统中本质安全要求的防爆场合。

③ PROFIBUS-FMS：定义了主站和从站之间的通信模型，主要用于自动化系统中车间级的数据交换。

3. 工业以太网（Industrial Ethernet）

工业以太网是目前工控领域中最为流行的网络技术，支持 TCP/IP、UDP、ISO 协议。工业以太网为 SIMATIC NET 提供了一个无缝集成到多媒体世界的途径。适用于大量数据的传输和长距离通信。在物理连接上，网络的传输介质可以是同轴电缆、双绞线、光纤和无线通信。在 SIMATIC 网络中，PLC 站须通过通信模块 CP（CP343－1、CP443－1 等）连接至工业以太网，PC 须通过网卡（西门子的 CP1613 或普通网卡）连接至以太网。工业以太网的通信速率为 10 Mbit/s 和 100 Mbit/s，目前已经有千兆工业以太网交换机，其推广指日可待。

4. 点对点连接（Point-to-Point）

点对点连接通信简称为 PtP 通信，使用带有 PtP 通信功能的 CPU 或通信处理器，可以与 PLC、计算机或别的带串口的设备通信，例如打印机、机器人控制器、调制解调器、扫描仪和条形码阅读器等。通信处理器 CP 340 或 CP 341 可以实现 S7 300 模块的点对点通信。S7 400 模块的点对点通信使用通信处理器 CP 440 或 CP 441 实现。

5. 传感器/执行接口（AS-Interface）

传感器/执行接口用于自动化系统最底层的设备级通信网络，它被专门设计用来连接二进制的传感器和执行器，每个从站的最大数据是 4 bit/s。

在本章中，我们将以 Siemens S7－300 的 PLC 为核心组建通信实例，重点介绍 MPI 网络、现场总线 PROFIBUS-DP 网络以及工业以太网的组建和应用。

## 9.2 MPI 通信

### 9.2.1 简介

MPI（Multi Point Interface）是指多点接口通信协议，通过它可组成一个小型 PLC 通信网络，实现 PLC 之间的少量数据交换，它不需要额外的硬件和软件就可网络化。每个 S7－300 CPU 都集成了 MPI 通信协议。MPI 的物理层是 RS－485。通过 MPI，PLC 可以同时与多个设备建立通信连接，这些设备包括编程器 PG 或运行 STEP7 的 PC、人机界面（HMI）及其他 SIMATIC S7、M7 和 C7。同时连接通信对象的个数与 CPU 型号有关。

对于连接站点少，通信速率要求不高的场合，MPI 是一种简单经济的通信方式。通过 MPI 组建的网络，其站点连接数量最多 32 个，每段最长为 50 m，增加 RS－485 中继器后可以扩大传输距离；两个中继器之间若没有其他节点，则最大可达 1 km。MPI 属于西门子保密协议，支持的厂家少。这里所说的支持，主要指各厂家的产品能通过 MPI 无缝连接到网络中。MPI 的组网成本较低且使用方便。

MPI 与后面章节要介绍的 PROFIBUS 协议的物理接口标准使用的都是 RS－485，属于串行、异步、半双工通信。传输速率一般为 19.2 Kbit/s、187.5 Kbit/s、1.5 Mbit/s 等。通常，在应用中不要改变 MPI 的通信速率，而且在整个 MPI 网络中通信速率必须保持一致，注意 MPI 站地址也不能冲突。

每个 S7－300/400－CPU 都支持 MPI 通信协议，并集成了 RS－485 接口，可以很方便地

组建 MPI 网络。网络上的每个节点都分配了一个唯一的 MPI 地址，用于站点的身份标识。编程设备（如 PC）、人机接口（如触摸屏）和 CPU 的默认地址分别为 0、1、2。用户在组建 MPI 网络的过程中，一般不要轻易改动这些默认值。

S7－200 的 CPU 可以通过自身集成的 RS－485 接口连接到 MPI 网络，实现与 S7－300/400 的通信。但是在 MPI 网络内 S7－200 是作为智能从站存在的，它们之间不能直接进行通信，一般是通过 S7－300/400 站作为中转。

### 9.2.2 通信分类

在 MPI 网络实现 PLC 到 PLC 之间通信有三种方式：

（1）全局数据包通信方式

如果在各个中央处理单元（CPU）之间相互交换少量数据，只关心数据的发送区和接收区，则可以采用全局数据块通信；这种通信方式只适合 S7－300/400 PLC 之间相互通信，应用范围不是很广。

（2）无组态连接通信方式

无组态的 MPI 通信适合于 S7－300、S7－400 和 S7－200 之间的通信，是一种应用广泛、经济的通信方式；通信双方都需要调用系统功能块 SFC65～SFC69 来实现，又分为双边编程通信方式和单边通信编程方式。

当实现 S7－300、S7－400 之间通信时，使用双边编程通信方式。双边通信方式在编程时通信的双方都需要调用通信块，一方调用发送块 SFC65（X-SEND）发送数据，另一方调用接收块 SFC66（X-RCV）来接收数据。

当实现 S7－300、S7－400 和 S7－200 之间通信时，使用单边编程通信方式。这种通信方式只在一方编写通信程序，是一种客户机与服务器的访问模式。S7－200 只能作为服务器，服务器一端无需编写程序，S7－300、S7－400 可以同时作为客户机和服务器，客户机调用 SFC 通信块访问服务器。SFC67（X-GET）用来将服务器指定数据区的数据读回并存放到本地的数据区，SFC68（X-PUT）用来将本地数据区中的数据写到服务器中指定的数据区。在单边编程通信方式中，服务器一方不需编程，也可以对收到的数据做处理或准备一些数据以便对方来读取，而组织数据的发送和读取任务由客户机承担。

（3）组态连接通信方式

组态连接通信方式只适合于 S7－300/400 以及 S7－400/400 之间的通信。S7－300/400 通信时，S7－300 只能作为服务器，S7－400 作为客户机对 S7－300 的数据进行读写操作；S7－400/400 通信时，S7－400 既可以作为服务器，也可以作为客户机。在 MPI 网络上调用系统功能块 SFB 时，数据包长度最大为 160B。SFB15（PUT）用来写数据到远程 CPU，SFB14（GET）用来读远程 CPU 数据。

调用系统功能块 SFB 与系统功能 SFC 的通信相比，每一包的发送接收数据量要大一些，但是要在硬件组态中建立连接表，并且同样要占用 S7－300 的通信资源，在满足通信要求的前提下，建议使用无组态连接的通信方式。

### 9.2.3 MPI 通信实例

本节用两个 S7－315－2PN/DP CPU 为通信对象来组建一个两节点的 MPI 通信网络，采

用无组态双边编程通信方式，详细介绍 MPI 通信网络的组态方法和通信程序的编写。

### 1. 硬件需求

① 两个 S7－315－2PN/DP（其他 300 系列 CPU 都可）。

② 插入 CP5611 网卡的 PC。

③ MPI 电缆及三个 DP 接头。

### 2. 物理连接

首先用工具制作带有三个接头的 PROFIBUS 电缆，并将两端接头上的拨码拨至 ON，中间的接头拨码拨至 OFF。然后利用这根 MPI 电缆将 PC、两个 CPU 315 建立 MPI 网络物理连接。

### 3. MPI 网络的硬件组态

首先，新建一个工程名为“COM_MPI_300－300”的项目，并在 SIMATIC Manager 中插入两个 300 站点“SIMATIC 300（1）”和“SIMATIC 300（2）”，根据订货号分别进行硬件组态。

（1）组态 SIMATIC 300（1）

“SIMATIC 300（1）”的硬件组态如图 9-1 所示。




图 9-1　SIMATIC 300（1）的硬件组态

双击图 9-1 的 2 号插槽内的 X1“MPI/DP”接口来配置 MPI 接口参数。需要说明的是，本例中所使用的 CPU 315－2PN/DP 本身集成了一个“PROFINET”接口和一个“MPI/DP”的复用接口，根据需要用户可以选择将“MPI/DP”接口配置成 MPI 还是 PROFIBUS 接口。本章要组建的是 MPI 网络，所以将“MPI/DP”配置成 MPI 接口。

在图 9-2 的“常规”选项卡内点击“MPI/DP”下拉菜单，选择“MPI”，将 X1 接口配置为 MPI 口；然后点击“属性”按钮。

在图 9-3 内可以设定 MPI 接口的参数：1 号框内设定站地址 2（可以设定为其他的地址）；2 号框内新建一个网络“MPI（1）”，可以通过 3 号框内的“属性”来设定网络的参数，如通信速率等，这里选择 187.5 Kbps，然后点击“确定”按钮。可以看到图 9-2 的“联网”状态变为“是”，表示组建了一个 MPI 网络，点击“确定”按钮。

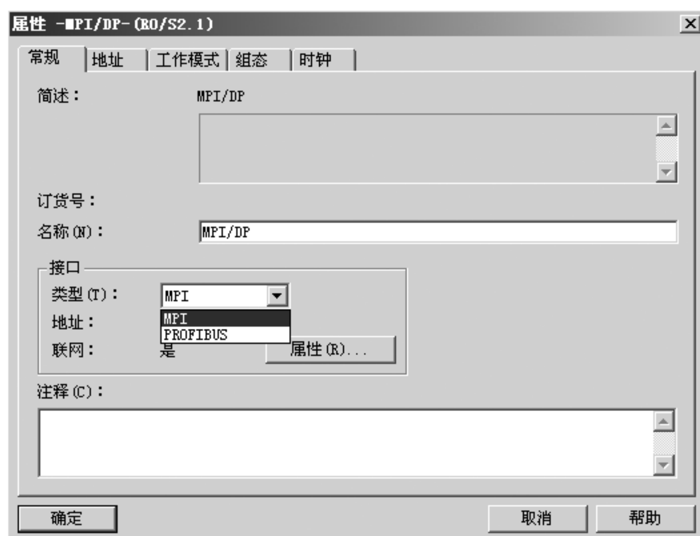



图 9-2　配置“MPI/DP”接口属性




图 9-3　设定 SIMATIC 300（1）MPI 接口的参数

至此，站点地址为 2 的站硬件组态完成。点击“编译保存”按钮进行编译保存。

（2）组态 SIMATIC 300（2）

本项目所使用的硬件订货号是一致的，所以组态的步骤、内容与组态“SIMATIC 300（1）”站一样，注意在设定该站的 MPI 地址的时候，不能和“SIMATIC 300（1）”站的 MPI 地址重复，这里设定“SIMATIC 300（2）”的 MPI 地址为 3，如图 9-4 所示，点击“确定”按钮，并对硬件组态进行编译保存。

**4. 在通信双方编写程序**

MPI 网络站点之间的数据传输可以采用双边编程和单边编程的方式实现。所谓双边编程，就是在通信双方均调用发送/接收系统块，进行数据的发送和接收；单边编程指的是仅在通信一方调用发送/接收系统块从而发起通信请求，而另一个节点则不发起通信请求，只是提供一些数据或什么工作也不做。两种方式可以通过调用 STEP7 内系统功能块来实现。




图 9-4　设定 SIMATIC 300（2）MPI 接口的参数

（1）系统功能 SFC65 和 SFC66

采用双边编程，通信双方的灵活性很大。程序可以通过调用系统功能 SFC65 和 SFC66 来完成。SFC65 即“X_SEND”的功能是向本地 S7 站以外的通信伙伴发送数据；而 SFC 66“X_RCV”则是接收本地 S7 外的通信伙伴的数据。SFC65 和 SFC66 的框图如图 9-5 所示，表 9-1 和表 9-2 是对这两个功能的引脚说明。

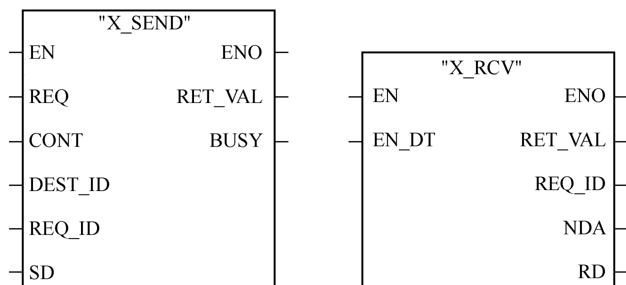



图 9-5　SFC65 和 SFC66 的框图

**表 9-1　X_SEND 引脚功能说明**

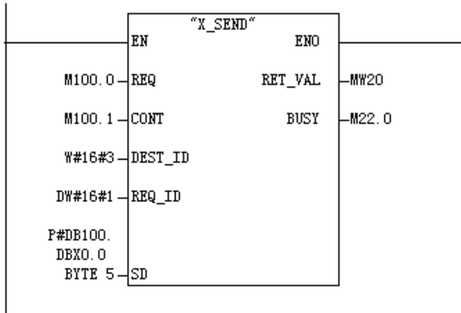
| 参数 | 数据类型 | 存储区域 | 说　　明 |
|---|---|---|---|
| EN | BOOL | I、Q、M 等 | 模块工作使能端，一般直接连到母线上 |
| REQ | BOOL | I、Q、M、D、L 常数 | 如果当前没有激活的作业调用 SFC65，则通过 REQ = 1 来触发数据发送作业。如果没有到通信伙伴的连接，则在数据传送开始之前首先建立连接 |
| CONT | BOOL | I、Q、M、D、L 常数 | 该端子决定数据传送作业结束之后是否保持建立与通信伙伴的连接 |
| REQ_ID | DWORD | I、Q、M、D、常数 | 用于标识发送的数据：1. 如果在一个发送 CPU 上调用几个 SFC65，并将数据传送到一个通信伙伴时则需不同的 REQ_ID；2. 如果几个 CPU 使用 SFC65 发送数据到同一个通信伙伴则需设定不同该参数 |
| DEST_ID | WORD | I、Q、M、D、L 常数 | 接收站的 MPI 地址 |
| SD | ANY | I、Q、M、D | 指向发送区的指针，如“P#M10.0 BYTE 100” |
| RET_VAL | INT | I、Q、M、D、L | 如果在函数执行过程中出错，则返回值包含相应的错误代码 |
| BUSY | BOOL | I、Q、M、D、L | BUSY = 1：发送还没有结束。BUSY = 0：发送已经结束或不存在已经激活的发送函数 |

表 9-2　X_RCV 引脚功能说明

| 参数 | 数据类型 | 存储区域 | 说　　明 |
|---|---|---|---|
| EN | BOOL | I、Q、M 等 | 模块工作使能端，一般直接连到母线上 |
| EN_DT | BOOL | I、Q、M、D、L、常数 | 通过数值 0，可以检查是否至少有一个数据块正在等待被输入到接收区。数值 1 复制队列中最早的数据块到 RD 指定的工作存储区域 |
| REQ_ID | DWORD | I、Q、M、D、L、常数 | “X_SEND”的作业标识符，用来区别接收到的各个作业；如果队列中没有数据块，则 REQ_ID 数值为 0 |
| NDA | BOOL | I、Q、M、D、L | 指示是否已有新数据到达。NDA = 0：队列中没有数据块；NDA = 1：队列至少包含了一个数据块（调用 SFC 66，EN_DT = 0）。队列中最早的数据块被复制到用户程序中（调用 SFC 66，EN_DT = 1） |
| RD | ANY | I、Q、M、D | 指向接收区的指针，如“P#M10. 0 BYTE 100” |
| RET_VAL | INT | I、Q、M、D、L | 如果在函数执行过程中出错，则返回值包含相应的错误代码 |

（2）编写“SIMATIC 300（1）”站程序

在“SIMATIC 300（1）”的块内首先插入组织块 OB1 和数据块 DB100，并在 DB100 内建立一个 20 个字节大小的数组，进行编译保存。在 OB1 内编写如图 9-6 所示程序。该程序的功能是通过两次调用“X_SEND”模块，向通信对方发送 2 个不同编号（由 REQ_ID 标记）的数据包。其中：



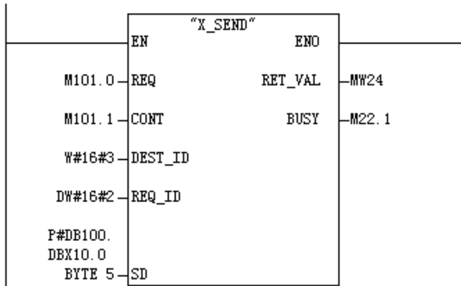




图 9-6　SIMATIC 300（1）站内的数据发送程序

通过 M100.0=1 来触发 SFC65“X_SEND”，向 3 号站发送 P#DB100.DBX0.0 开始的 5 个字节内的数据，该数据块的 REQ_ID 编号为 1。

通过 M101.0=1 来再次触发 SFC65“X_SEND”，向 3 号站发送 P#DB100.DBX10.0 开始的 5 个字节内的数据，该数据块的 REQ_ID 编号为 2。

因为“X_SEND”是在 REQ=1 的情况下触发数据发送，且两个数据包经由同一通道发送，故两者不能同时触发，为了避免这种情况的发生，可以采用如图 9-7 所示程序来控制 M100.0 和 M101.0。

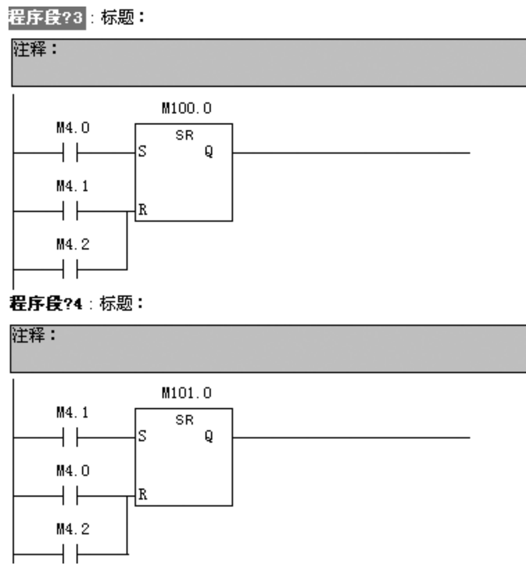



图 9-7 控制 X_SEND 模块的激活

（3）编写“SIMATIC 300（2）”站程序

在“SIMATIC 300（2）”（MPI 站地址为 3）站的块内首先插入组织块 OB1 和数据块 DB102，并在 DB102 内建立一个 20 个字节大小的数组，进行编译保存。

在 OB1 内编写如图 9-8 所示程序。该程序的功能是通过调用“X_RCV”读取缓冲区队列中的数据块，并存储到 RD 指定的存储区内。其中：

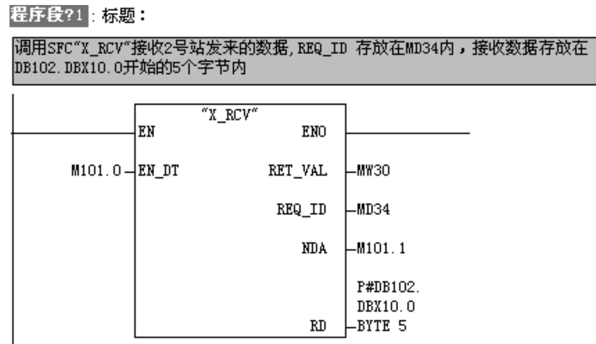



图 9-8 检查并读取站外发来的数据

当 M101.0(EN_DT)=0 时，SFC66“X_RCV”模块检查通信缓冲区队列是否有新数据块，如果有则 M101.1(NDA)=1，且队列中最早存在的数据块的 REQ_ID 号存储在 MD34

中；若没有则 NDA（M1.1）为 0。

由 M101.0 = 1 来触发 SFC66“X_RCV”读取 2 号站发来的数据并存储到 RD 指定的存储区内，本例中指定的是从 P#DB102.DBX10.0 开始的 5 个字节；这里要特别注意的是，RD 指定的存储区的大小一定要大于等于 SFC65“X_SEND”中 SD 所指定的发送区，否则容易出错。

如果 RD 指定的存储区中的数据没有被及时处理，这时若 M101.0 又一次触发 SFC“X_RCV”读取数据，则 RD 中的数据会被后来的数据覆盖。为了避免这种情况发生，可以作如下处理：根据数据块 REQ_ID 号的不同利用 SFC20（BLKMOV）将数据分别存入不同的处理区，程序如图 9-9 所示。

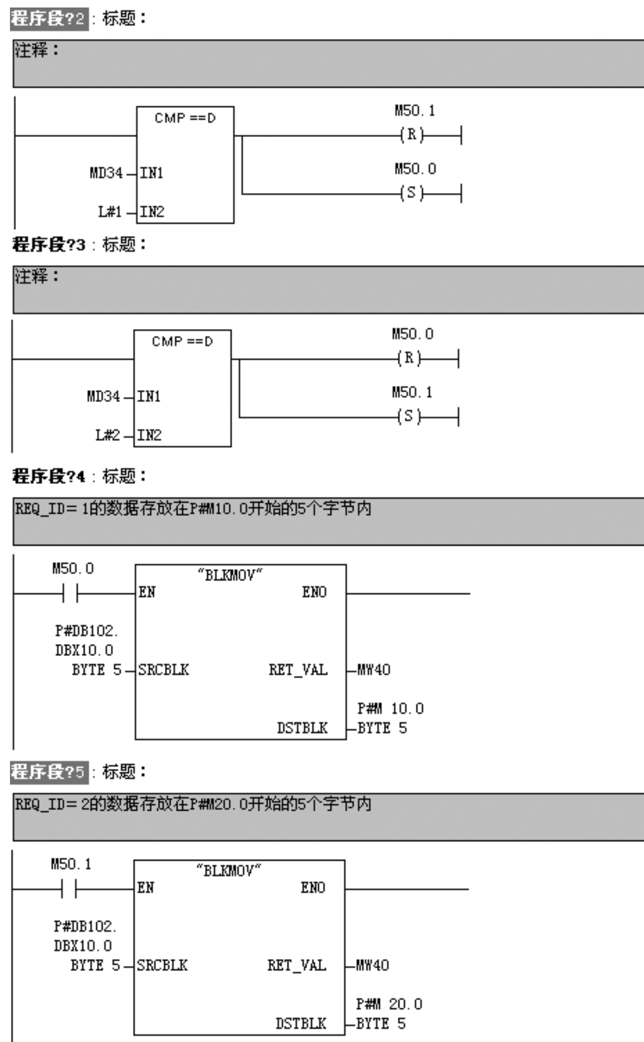



图 9-9　根据数据块标号读取数据

### 5. 项目下载和监控

（1）项目下载

可以使用 3.6 节中所介绍的方法，将两个站点“SIMATIC 300（1）”和“SIMATIC 300（2）”的硬件组态和程序块都下载到各自的 PLC 内，如图 9-10 所示。

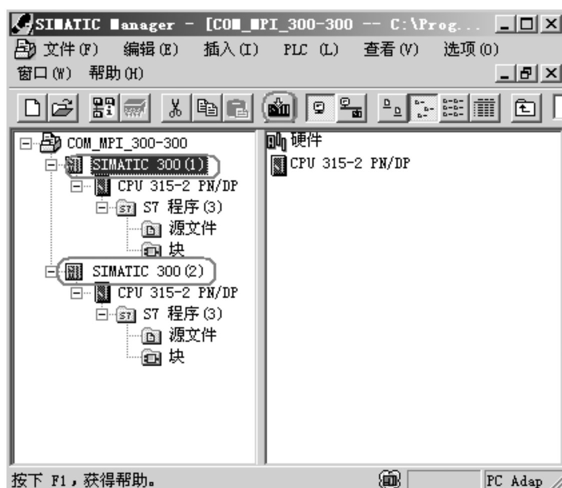



图 9-10 下载项目硬件组态和程序

（2）监控设置

本例也采用 MPI 方式来进行程序运行状态的监控，这需要 MPI 紫色电缆和 MPI 通信卡（比如 CP5611）的支持。

在“Set PG/PC 接口”内修改访问路径为“S7ONLINE（STEP7）→CP5611（MPI）”，如图 9-11 所示。单击“Diagnostics...”，可以监控每个站点是否连通。

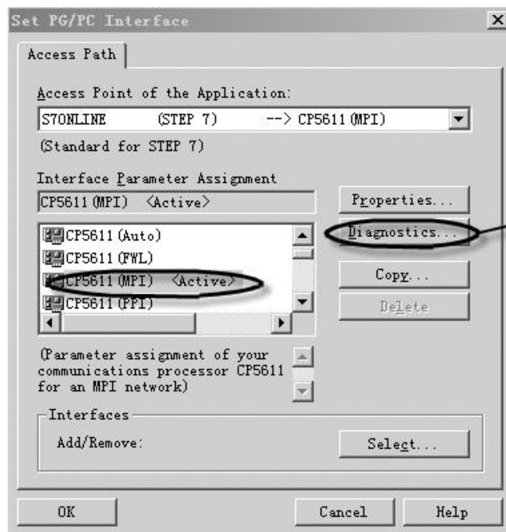



图 9-11 监控设置

（3）监控测试

本项目做了如下测试，在 2 号站中发送两个不同 REQ_ID 号的数据块，在 3 号站内将这两个数据块存储在不同的数据区内。

首先在“SIMATIC 300（1）”的块内插入一个 DB100 的数据块，并以数组的形式开辟两个各为 5B 大小的数据块，分别为 DB100.DBX0.0 BYTE 5（REQ_ID = 1）和

DB100. DBX10. 0 BYTE 5（REQ_ID =2），保存下载。

然后在“SIMATIC 300（2）”的块内插入一个 DB102 的数据块，分配大小为至少 5B 的存储区，即 DB102. DBX10. 0 BYTE 5。保存下载。

在“SIMATIC 300（1）”和“SIMATIC 300（2）”中分别插入变量表 VAT_1 和 VAT_2，输入如图 9-12 和图 9-13 所示内容，并观察监控的效果。

变量 - [VAT_1 -- @COM_MPI_300-300\SIMATIC 300(1)\CPU 315-2 P...

表格(T) 编辑(E) 插入(I) PLC(L) 变量(A) 查看(V) 选项(O) 窗口(W) 帮助(H)

| | 地址 | 符号 | 显示格式 | 状态值 | 修改数值 |
|---|---|---|---|---|---|
| 1 | //2号站的控制端 | | | | |
| 2 | //触发发送M4.0 REQ_ID=1 | | | | |
| 3 | M 4.0 | | BOOL | true | true |
| 4 | MW 20 | | HEX | W#16#7002 | |
| 5 | MW 24 | | HEX | W#16#80C0 | |
| 6 | DB100.DBB 0 | | HEX | B#16#01 | B#16#01 |
| 7 | DB100.DBB 1 | | HEX | B#16#02 | B#16#02 |
| 8 | DB100.DBB 2 | | HEX | B#16#38 | B#16#38 |
| 9 | DB100.DBB 3 | | HEX | B#16#12 | B#16#12 |
| 10 | DB100.DBB 4 | | HEX | B#16#29 | B#16#29 |
| 11 | //触发发送M4.1 REQ_ID=2 | | | | |
| 12 | M 4.1 | | BOOL | false | false |
| 13 | DB100.DBB 10 | | HEX | B#16#06 | B#16#06 |
| 14 | DB100.DBB 11 | | HEX | B#16#25 | B#16#25 |
| 15 | DB100.DBB 12 | | HEX | B#16#08 | B#16#08 |
| 16 | DB100.DBB 13 | | HEX | B#16#09 | B#16#09 |

COM_MPI_300-300\SIMATIC 300(1)\...\S7 程序(3)　RUN

图 9-12　发送 REQ =1 数据块

变量 - [VAT_1 -- @COM_MPI_300-300\SIMATIC 300(2)\CPU 315-2 ...

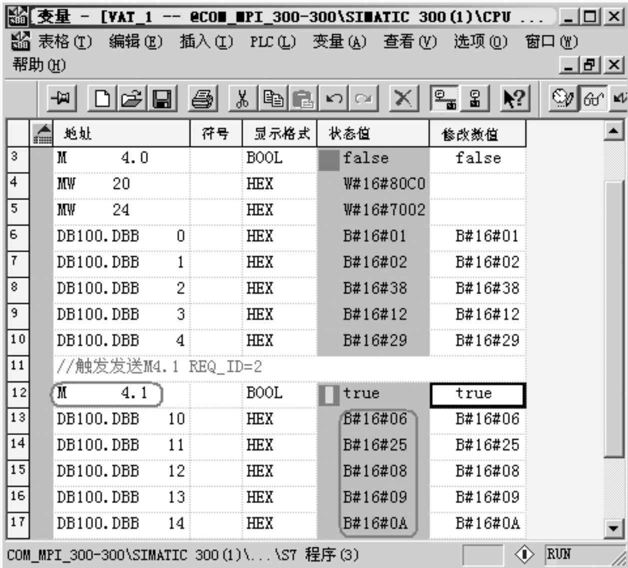
表格(T) 编辑(E) 插入(I) PLC(L) 变量(A) 查看(V) 选项(O) 窗口(W) 帮助(H)

| | 地址 | 符号 | 显示格式 | 状态值 | 修改数值 |
|---|---|---|---|---|---|
| 1 | //3号站的控制 | | | | |
| 2 | M 101.0 | | BOOL | true | true |
| 3 | //REQ_ID | | | | |
| 4 | MD 34 | | HEX | DW#16#00000000 | |
| 5 | //NDA | | | | |
| 6 | M 101.1 | | BOOL | false | |
| 7 | MW 24 | | HEX | W#16#0A00 | |
| 8 | //REQ_ID=1的数据块 | | | | |
| 9 | MB 10 | | HEX | B#16#01 | |
| 10 | MB 11 | | HEX | B#16#02 | |
| 11 | MB 12 | | HEX | B#16#38 | |
| 12 | MB 13 | | HEX | B#16#12 | |
| 13 | MB 14 | | HEX | B#16#29 | |
| 14 | //REQ_ID=2的数据块 | | | | |
| 15 | MB 20 | | HEX | B#16#00 | |
| 16 | MB 21 | | HEX | B#16#00 | |
| 17 | MB 22 | | HEX | B#16#00 | |
| 18 | MB 23 | | HEX | B#16#00 | |
| 19 | MB 24 | | HEX | B#16#0A | |

COM_MPI_300-300\SIMATIC 300(2)\...\S7 程序(3)　RUN

图 9-13　接收 REQ =1 数据块

运行程序。首先在 VAT_1 中修改 REQ_ID1 = 1 的数据块的值，并修改 VAT_1 的 M4.0 值为 1，如图 9-12 所示。可以看到图 9-13 中 VAT_2 的 MB10 ~ MB14 接收到该数据；在 VAT_1 中修改 REQ_ID1 = 2 的数据块的值，并修改 VAT_1 的 M4.1 值为 1，如图 9-14 所示。可以看到图 9-15 中 VAT_2 的 MB20 ~ MB24 接收到相应的数据。




| | 地址 | 符号 | 显示格式 | 状态值 | 修改数值 |
|---|---|---|---|---|---|
| 3 | M 4.0 | | BOOL | false | false |
| 4 | MW 20 | | HEX | W#16#80C0 | |
| 5 | MW 24 | | HEX | W#16#7002 | |
| 6 | DB100.DBB 0 | | HEX | B#16#01 | B#16#01 |
| 7 | DB100.DBB 1 | | HEX | B#16#02 | B#16#02 |
| 8 | DB100.DBB 2 | | HEX | B#16#38 | B#16#38 |
| 9 | DB100.DBB 3 | | HEX | B#16#12 | B#16#12 |
| 10 | DB100.DBB 4 | | HEX | B#16#29 | B#16#29 |
| 11 | //触发发送M4.1 REQ_ID=2 | | | | |
| 12 | M 4.1 | | BOOL | true | true |
| 13 | DB100.DBB 10 | | HEX | B#16#06 | B#16#06 |
| 14 | DB100.DBB 11 | | HEX | B#16#25 | B#16#25 |
| 15 | DB100.DBB 12 | | HEX | B#16#08 | B#16#08 |
| 16 | DB100.DBB 13 | | HEX | B#16#09 | B#16#09 |
| 17 | DB100.DBB 14 | | HEX | B#16#0A | B#16#0A |

图 9-14　发送 REQ = 2 数据块




| | 地址 | 符号 | 显示格式 | 状态值 | 修改数值 |
|---|---|---|---|---|---|
| 7 | MW 24 | | HEX | W#16#0A00 | |
| 8 | //REQ_ID=1的数据块 | | | | |
| 9 | MB 10 | | HEX | B#16#00 | |
| 10 | MB 11 | | HEX | B#16#00 | |
| 11 | MB 12 | | HEX | B#16#38 | |
| 12 | MB 13 | | HEX | B#16#12 | |
| 13 | MB 14 | | HEX | B#16#29 | |
| 14 | //REQ_ID=2的数据块 | | | | |
| 15 | MB 20 | | HEX | B#16#06 | |
| 16 | MB 21 | | HEX | B#16#25 | |
| 17 | MB 22 | | HEX | B#16#08 | |
| 18 | MB 23 | | HEX | B#16#09 | |
| 19 | MB 24 | | HEX | B#16#0A | |
| 20 | | | | | |

图 9-15　接收 REQ = 2 数据块

## 9.3　PROFIBUS 现场总线通信

### 9.3.1　简介

随着通信技术的发展，结构简单、成本低廉、可远程传输的串行通信方式在工业控制领

域得到了广泛的应用。具有串行通信接口的设备如果采用统一的通信协议，便可以通过一对双绞线来实现现场信号的传输。现场总线连接方式如图 9-16 所示，基于此，现场总线的概念在 1984 年得到正式提出并逐渐被广泛应用。现场总线实现了数字和模拟输入/输出模块、智能信号装置和过程调节装置与 PLC 和 PC 之间的数据传输，把 I/O 通道分散到实际需要的现场设备附近，从而使整个系统的工程费用、装配费用、硬件成本、设备调试和维修成本减少到最少。控制器与现场设备可以实现双向的数字通信，克服了模拟信号精度不高、抗干扰能力差的缺点，提高了系统的可靠性。

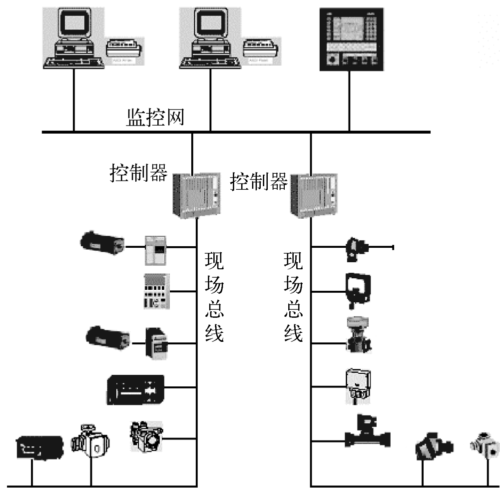



图 9-16　现场总线连接方式

标准化的现场总线具有开放的通信接口、透明的通信协议。经过 12 年的讨论，国际电工委员会（IEC）终于在 1999 年底通过了一个 IEC61158 的现场总线标准，这个标准容纳了 8 种互不兼容的总线协议。后来又经过不断讨论和协商，在 2003 年 4 月，IEC61158 Ed. 3 现场总线标准第 3 版正式成为国际标准，规定 10 种类型的现场总线，其中包括 FF 现场总线、LONWORKS 总线、PROFIBUS 现场总线、CANBUS 现场总线、ProfiNet 现场总线等。每种总线协议都有其特有的应用领域和支持背景；有的成为了一个国家或者一个地区的标准；目前是谁也取代不了谁。IEC61158 国际标准中的 PROFIBUS 现场总线也是德国标准（DIN19245）和欧洲标准（EN50170）。在 2001 年 PROFIBUS 被定为中国的国家标准 JB/T103010. 3 -2001。

### 9. 3. 2　协议类型分类

从用户角度考虑，PROFIBUS 有 PROFIBUS-DP、PROFIBUS-FMS 和 PROFIBUS-PA 三种通信协议类型。

① PROFIBUS-DP（Decentralized Periphery，分布式外围设备），用于分散外设与控制设备间的高速数据传输，适用于加工自动化领域，可以取代 4 ~ 20 mA 的模拟信号传输。PROFIBUS-DP 使用了 ISO/OSI 模型的第 1 层（物理层）和第 2 层（数据链路层），使网络获得较高的传输速率。PROFIBUS-DP 特别适合于 PLC 与现场级分布式 I/O（如 Siemens 的 ET200）设备之间的通信。

② PROFIBUS-FMS（Fieldbus Message Specification，现场总线报文规范），适用于纺织、

楼宇自动化、可编程序控制器和低压开关等。除了 OSI 的第 1 层和第 2 层，PROFIBUS-FMS 还使用了第 7 层，即应用层，因此该协议向用户提供了功能很强的通信服务。主要用于车间级的不同供应商的自动化之间传输数据。

③ PROFIBUS-PA（Process Automation，过程自动化）是专为过程自动化设计的总线类型，使用的是扩展的 PROFIBUS-DP 协议，此外还描述了现场设备行为的 PA 行规。其传输技术使用的是 IEC1158－2，确保了本质和系统的稳定性，并通过总线对现场设备供电。PROFIBUS-PA 广泛应用于化工和石油生产等领域。

### 9.3.3 PROFIBUS-DP 通信及分类

在这三种 PROFIBUS 协议中，PROFIBUS-DP 解决的是分布式现场设备与控制器之间的数据交换，应用范围最为广泛。所以，本书将重点介绍基于 PROFIBUS-DP 的通信网络的组建与应用。

PROFIBUS-DP 和 PROFIBUS-FMS 使用的是 RS－485 传输技术，传输介质可以采用屏蔽双绞线和光纤等。使用屏蔽双绞线的传输速率有 9.6 Kbit/s、19.2 Kbit/s、93.75 Kbit/s、187.5 Kbit/s、500 Kbit/s、1500 Kbit/s、12 000 Kbit/s。随着通信速率的增加，传输距离也相应地降低为 1200 m、1200 m、1200 m、1000 m、400 m、200 m、100 m。

基于 PROFIBUS-DP 的网络可以构建主从（MS）模式和直接数据交换（DX）通信方式。

直接数据交换方式通信在工程实际中应用较少，主从模式是 PROFIBUS 网络的典型结构。PROFIBUS 主站可以是带有集成 DP 口的 CPU，或者用 CP342－5 扩展的 S7－300 站、IM467、CP443－5Extend 扩展的 S7－400 站，上位机中插有 CP5411、CP5511、CP5611 等通信卡，也可以作为 PROFIBUS-DP 主站，这些通信卡加入 PROFIBUS 驱动程序就可以当中 PROFIBUS 网卡并支持 PROFIBUS 协议。PROFIBUS 从站有 ET200 系列、调速装置、S7－200/300/400 站及第三方设备等。

根据数据传输速率、数据的吞吐量等相关要求，可以组建不同的 PROFIBUS-DP 网络以实现不打包通信和打包通信。

① 打包通信需要调用系统功能（SFC）。STEP7 提供了两个系统功能（SFC15 和 SFC14）来完成数据的打包和解包功能。

② 不打包通信可直接利用传送指令实现数据的读写，但是每次最大只能读写 4 个字节（双字）。

### 9.3.4 PROFIBUS-DP 通信实例

不打包通信每次传输的数据最大为 4 个字节，若想一次传送更多的数据，则应该采用打包方式的通信。本节用三个 S7－315－2PN/DP CPU 为通信对象来组建一个 PROFIBUS-DP 通信网络，采用打包通信方式，详细介绍网络的组态方法和通信程序的编写。其物理连接方法参考 MPI 通信实例。

打包通信需要调用系统功能（SFC）。STEP－7 提供了两个系统功能 SFC15 和 SFC14 来完成数据的打包和解包功能。

#### 1. PROFIBUS-DP 网络的硬件组态

首先，新建一个工程名为“PROFIBUS300－300（ALL）”的项目，并在 SIMATIC Man-

ager 中插入三个 300 站点并重命名为“SIMATIC 300（M）”、“SIMATIC 300（S1）”和“SIMATIC 300（S2）”，其中“SIMATIC 300（M）”是主站，“SIMATIC 300（S1）”和“SIMATIC 300（S2）”是两个从站，如图 9-17 所示，并根据订货号分别进行硬件组态。

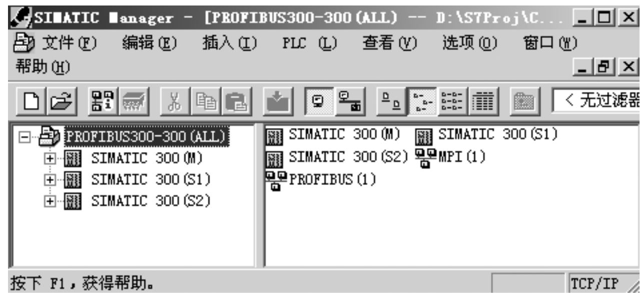



图 9-17　插入 1 个主站 2 个从站并重命名

接下来应该分别对主从站进行网络配置；原则上讲，先配置从站更便于网络的组建。

（1）组态 SIMATIC 300（S1）站的网络

在插入 CPU 的时候，系统会提示是否建立以太网和 PROFIBUS 网，均可以点击“取消”按钮。在图 9-1 中双击 2 号插槽里的 X1“MPI/DP”接口，出现如图 9-18 所示“属性 - MPI/DP”配置对话框，在 1 号框内设定 MPI/DP 接口为 PROFIBUS 接口；然后点击 2 号框“属性”按钮，进一步配置 PROFIBUS 参数：新建一条 PROFIBUS 电缆和 PROFIBUS 网络中从站的站地址（这里设为 3），在网络属性内选择网络的通信速率（1.5 Mbit/s）和 DP 行规。




图 9-18　“属性 - MPI/DP”配置对话框

因为现在要配置的是 PROFIBUS 网络上的一个从站，在如图 9-18 的“工作模式”选项卡内，选择“DP 从站”，如图 9-19 所示。DP 从站下面的信息包含诊断的地址、测试、主站的一些情况（如果主站已经配置完成）。

设定好操作模式后，还应该配置从站与主站的通信区，单击如图 9-18 中的“组态”选项卡来组态该从站与主站的通信区。在随后出现的对话框内点击“新建”按钮，出现如图 9-20 所示的对话框。

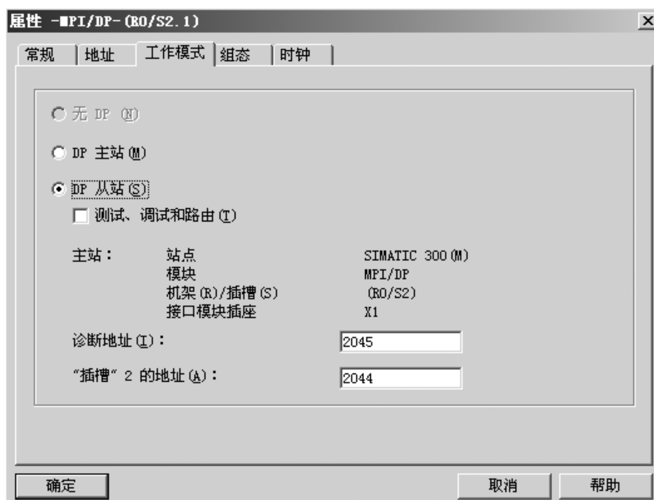



图 9-19　设定从站操作属性




图 9-20　SIMATIC 300（S1）从站输入通信区的设定

1 号框内可以设置通信输入/输出区，这里选择“输入”，即要创建的是数据接收区。在地址栏设置通信区的起始地址，默认是 0，这里设置起始地址为 10，即从站的输入区起始地址为 10。2 号框内可以设置通信区的长度、单位、数据的一致性，这里设置输入区的长度为 8，单位为“字节”，一致性为“全部”。设定完成后单击“确定”按钮。

采用上述相同的办法可以再新建该从站的数据发送区，即在图 9-20 的“地址类型”内选择“输出”，其他设置都与接收区相同。单击“确定”按钮。

配置好 SIMATIC 300（S1）站的通信区以后，通过点击“确定”按钮，在硬件管理器界面下点击“编译保存”按钮，对从站的配置编译保存。

（2）组态 SIMATIC 300（S2）站的网络

SIMATIC 300（S2）站建立的也是 PROFIBUS（1）网络，通信速率为 1.5 Mbit/s，行规

为 DP，站地址设为 5。

配置 SIMATIC 300（S2）站与主站的通信区与 SIMATIC 300（S1）站的通信区配置方法一样。如图 9-21 所示，SIMATIC 300（S2）站的输入/输出通信区起始地址本例都设为 20，通信区大小为 8 个字节。点击“确定”按钮，在硬件管理器界面下点击“编译保存”按钮，对从站的配置编译保存。




图 9-21 SIMATIC 300（S2）站输入通信区的设定

（3）组态 SIMATIC 300（M）站的网络

做好这个站的基本硬件组态后，双击主站 2 号插槽内的“MPI/DP”来配置主站的 PROFIBUS 网络参数：在图 9-18 中配置“MPI/DP”集成接口为 PROFIBUS，并设定 PROFIBUS (1) 网络中主站的站地址为 2，通信速率为 1.5 Mbit/s（同一个网络内的所有站点的通信速率应该相同）；在图 9-19 中设定“DP 主站”工作模式。

设定完成上述内容后，依次点击“确定”按钮。在“硬件组态”界面内可以看到“MPI/DP”后面拖出一条 PROFIBUS 电缆。此时，需要将刚才配置好的两个从站挂接到主站的 PROFIBUS-DP 的电缆上。在右侧的硬件模块目录树内依次选择 PROFIBUS-DP/Configured Station，将框内的 CPU 31x 拖至左侧的 PROFIBUS 电缆处，如图 9-22 所示。

在拖曳的过程中出现如图 9-23 所示对话框，单击“连接”按钮，将 SIMATIC 300（S1）站（地址为 3）连接到 PROFIBUS（1）网络上。重复上面的步骤，将 SIMATIC 300（S2）站（地址为 5）也连接到 PROFIBUS（1）网络上。连接后的结果如图 9-24 所示。

在图 9-24 内分别双击两个站的图标，从而进一步配置主站与两个从站的通信区。

在配置主站与 SIMATIC 300（S1）站的通信区对话框中（如图 9-25 所示），数据的“一致性”属性选择“全部”。通信区大小与从站的发送区的大小相同，也是 8 个字节；输入区地址设为 12；输出区的起始地址也是 12（这样的设置完全为了便于记忆，输入区和输出区起始地址可以不同）；设定完成后依次点击“确定”按钮。

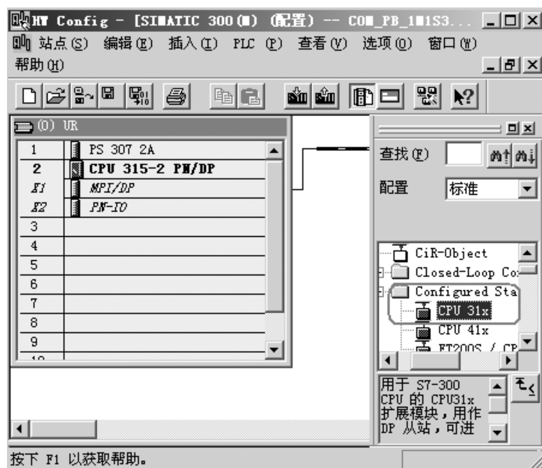



图 9-22　寻找配置好的从站

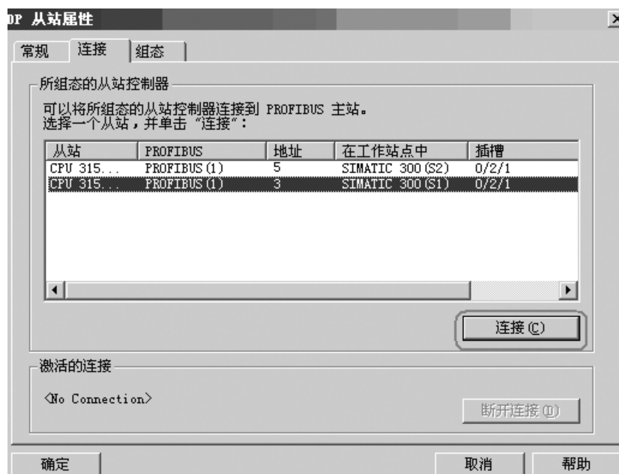



图 9-23　连接从站到 PROFIBUS – DP 电缆

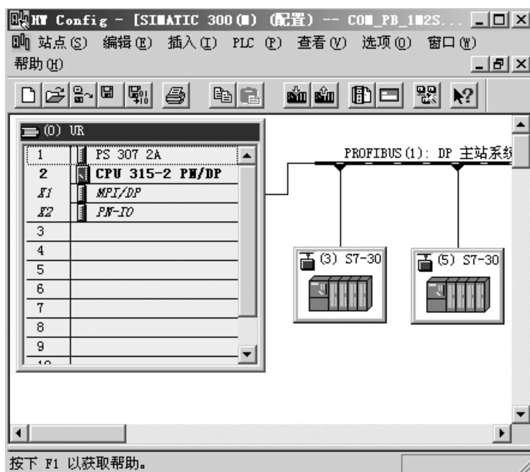



图 9-24　两个从站被连接到 PROFIBUS（1）网络




图 9-25　主站与 SIMATIC 300（S1）站的通信区配置

在配置主站与 SIMATIC 300（S2）站的通信区对话框中（如图 9-26 所示），发送（接收）通信区大小与从站的发送（接收）区的大小相同，也是 8 个字节；输入区地址设为 22；输出区的起始地址也是 22；设定完成后依次点击“确定”按钮。




图 9-26　主站与 SIMATIC 300（S2）站的通信区配置

图 9-27 和图 9-28 分别清楚地给出了主站与两个从站的通信区的通信方式（MS）、输入、输出通信区的起始地址、通信区的大小、一致性属性等。用户应该记住这些配置参数，不要混淆，理清数据之间的流向，以免在编程的时候发生混乱，导致通信难以成功。

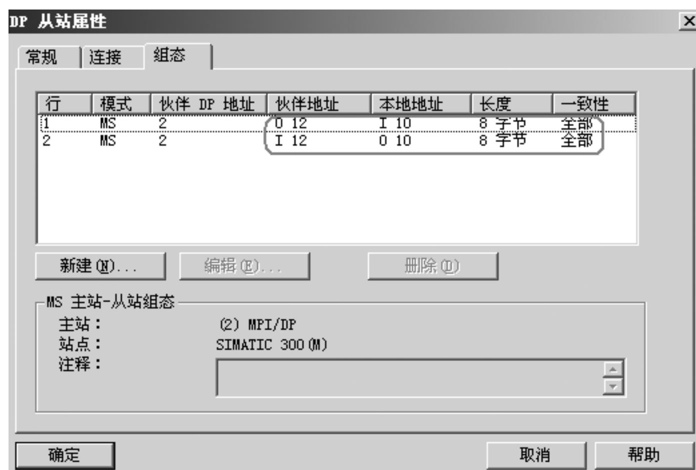



图 9-27　主站与 SIMATIC 300（S1）站的通信区




图 9-28　主站与 SIMATIC 300（S2）站的通信区

回到硬件组态界面，单击“编译保存”按钮，编译保存主站的硬件组态。在设置好下载路径后，将主站和从站的硬件组态分别下载到各自的 PLC 内。

**2. 软件编程**

PROFIBUS 主从（MS）模式网络都是由主站采用轮询的方式与从站实现通信。主站轮询到哪个从站，哪个从站才有发言权；从站之间不能直接进行通信，必须经由主站的参与。

主站和从站可以分别调用 SFC15、SFC14，实现双向通信，也可以在一边单独调用 SFC15，另一边单独调用 SFC14，实现单向通信。如果要使用 DB 块存储数据，还必须在项目管理器内建立所使用的 DB 块，并分配相应大小的存储区。

本例中采用的方案是在 SIMATIC 300（S1）站内发送 8 个字节的数据包给主站，主站接收到该数据包后解压缩，并再次打包发送给 SIMATIC 300（S2）站，SIMATIC 300（S2）站接收后解压缩并存储在内存区。

（1）SFC15 和 SFC14 简介

SFC15 的功能是对要发送的数据进行打包，并传给发送区，SFC14 的功能是对接收的数据进行解包，并转存至系统内存区。SFC15 和 SFC14 的引脚功能分别如表 9-3 和表 9-4 所示。

表 9-3　SFC15 引脚功能

| | | |
|---|---|---|
| "DPWR_DAT"<br>EN　ENO<br>LADDR　RET_VAL<br>RECORD | EN | 模块执行使能端 |
| | LADDR | 通信区的起始地址，以 W#16#格式给出 |
| | RECORD | 待打包的数据存放区，以指针形式给出 |
| | RET_VAL | 返回的状态值，字型数据 |
| | ENO | 输出使能 |

表 9-4　SFC14 引脚功能

| | | |
|---|---|---|
| "DPRD_DAT"<br>EN　ENO<br>LADDR　RET_VAL<br>RECORD | EN | 模块执行使能端 |
| | LADDR | 通信区的起始地址，以 W#16#格式给出 |
| | RECORD | 解包后数据存放区，以指针形式给出 |
| | RET_VAL | 返回的状态值，字型数据 |
| | ENO | 输出使能 |

（2）SIMATIC 300（S1）从站侧的编程

在 SIMTIC Manager 的“块”内插入 OB35 块，插入后结果如图 9-29 所示。新建两个数据块 DB1、DB2，分配 8 个字节的内存区，如图 9-30 所示。这里在 DB1 里设置了一个 8 个单元的数组，数组元素为字节型，DB1 的大小为 8 个字节。用同样的方法配置 DB2 的大小也为 8 个字节（大小可任意选择）。建立 DB 块后，必须保存。

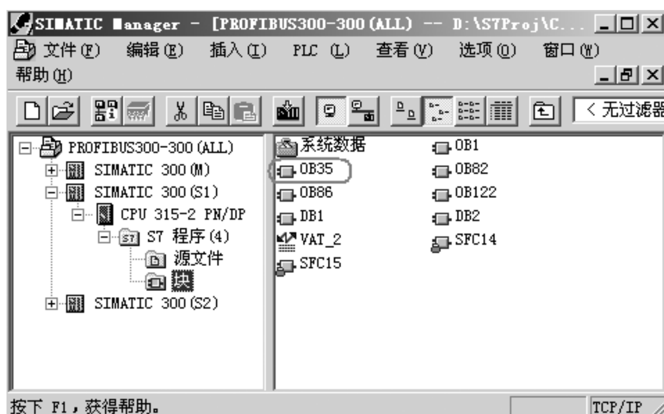



图 9-29　插入 OB35 块

在图 9-29 中，也为从站插入 3 个组织块，分别为 OB82、OB86 和 OB122。它们的作用主要是保证通信正常进行。具体的说明可以参见相关的手册。做好这些准备工作，就可以打开从站的 OB35 块，编写通信程序，具体程序如图 9-31 所示。

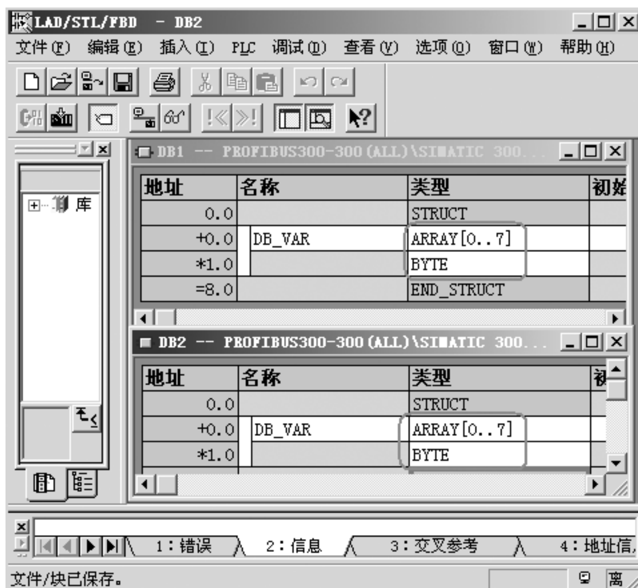



图 9-30　DB 块的存储大小设置

其中：W#16#A 是发送缓冲区起始地址（十进制为 10）；DB1. DB_VAR 是待发送数据的存储区；MW2 用来存储 SFC15 执行后的一些返回信息，通过该返回信息可以判断通信情况。

这段程序的功能是将 DB1. DB_VAR 内的数据打包发送给主站，程序按照 OB35 的中断时间周期地被执行。当然用户也可以将程序编写在 OB1、FC、FB 等块内，实现有条件的调用发送。

（3）SIMATIC 300（S2）从站侧的编程

与 SIMATIC 300（S1）站类似，在“块”内新建 DB1 数据块并同样分配 8 个字节的内存区。在“块”内插入 OB35 块。OB35 内编写如图 9-32 所示的程序。

OB35 : "Cyclic Interrupt"

注释:

程序段?1 : 标题:

将DB1内的数据写入发送缓冲区

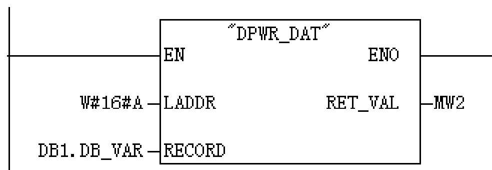



图 9-31　SIMATIC 300（S1）站的编程

OB35 : "Cyclic Interrupt"

注释:

程序段?1 : 标题:

读取主站发来的数据并存入DB1内

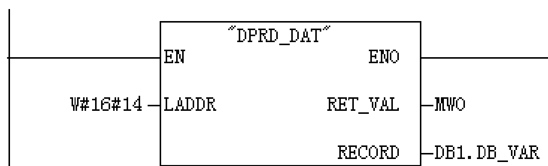



图 9-32　SIMATIC 300（S2）站的编程

其中：W#16#14 是该站接收缓冲区起始地址（十进制为 20）；DB1. DB_VAR 是接收数据的存储区；MW0 用来存储 SFC14 执行后的一些返回信息，通过该返回信息可以判断通信情况。

这段程序的功能是将主站发来的数据解包，并存储在 DB1. DB_VAR 内。

（4）主站侧的编程

本例中主站内不建立 DB 块，使用中间存储区 M 来实现数据读写。图 9-33 就是主站侧的程序。它的功能是将 MB50 开始的 8 个字节内的数据进行打包并发送给 SIMATIC 300（S2）站，而将 SIMATIC 300（S1）站发来的数据读取进来并解包存储在 MB50 开始的 8 个字节内。

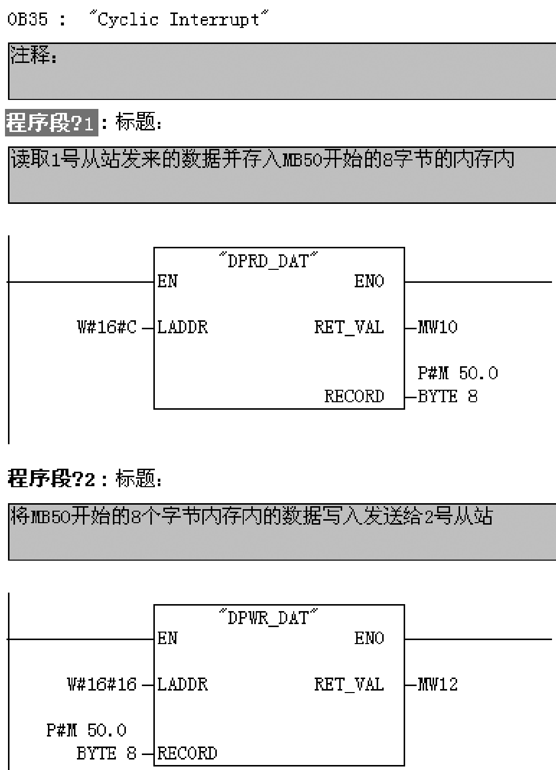



图 9-33　主站侧的程序

### 3. 项目下载和监控

将主、从站侧的硬件组态和程序分别下载到各自的 PLC 内，使用 PROFIBUS - DP 的紫色电缆连接好网络，如果硬件连接线路及程序编写没有问题（PLC 的面板上 SF 和 BF 的指示灯不亮），就可以通信了。具体的下载方法可以参照 3. 6 节所述的相关内容。

在主、从站的块内分别建立变量表，并在变量表内输入如图 9-34 所示的内存变量。

硬件组态和程序一旦正确下载，且已经完全重起动，就可以在变量表内对变量进行修改和监控。打开监控开关（眼镜状按钮），给 SIMATIC 300（S1）站的 DB1 赋值，通过观察图 9-34 所示的结果可以发现 SIMATIC 300（S2）站已经接收到 SIMATIC 300（S1）站发来的数据了。

变量 - [VAT_2 -- @COM_PB_1M2S_300-300(ALL)\SIMATIC 300...

表格(T) 编辑(E) 插入(I) PLC(L) 变量(A) 查看(V) 选项(O) 窗口(W) 帮助(H)

| | 地址 | | 符号 | 显示格式 | 状态值 | 修改数值 |
|---|---|---|---|---|---|---|
| 1 | //S1从站欲发送的数据 | | | | | |
| 2 | DB1.DBB | 0 | | HEX | B#16#11 | B#16#11 |
| 3 | DB1.DBB | 1 | | HEX | B#16#22 | B#16#22 |
| 4 | DB1.DBB | 2 | | HEX | B#16#33 | B#16#33 |
| 5 | DB1.DBB | 3 | | HEX | B#16#44 | B#16#44 |
| 6 | DB1.DBB | 4 | | HEX | B#16#55 | B#16#55 |
| 7 | DB1.DBB | 5 | | HEX | B#16#66 | B#16#66 |
| 8 | DB1.DBB | 6 | | HEX | B#16#77 | B#16#77 |
| 9 | DB1.DBB | 7 | | HEX | B#16#88 | B#16#88 |
| 10 | | | | | | |

COM_PB_1M2S_300-300(ALL)\SIMATIC 300(S1)\...\S7 程序(4 RUN

a)

变量 - [VAT_1 -- @COM_PB_1M2S_300-300(ALL)\SIMATIC 300...

表格(T) 编辑(E) 插入(I) PLC(L) 变量(A) 查看(V) 选项(O) 窗口(W) 帮助(H)

| | 地址 | | 显示格式 | 状态值 | 修改数值 |
|---|---|---|---|---|---|
| 1 | //主站接收的S1站发来的数据 | | | | |
| 2 | MB | 50 | HEX | B#16#11 | |
| 3 | MB | 51 | HEX | B#16#22 | |
| 4 | MB | 52 | HEX | B#16#33 | |
| 5 | MB | 53 | HEX | B#16#44 | |
| 6 | MB | 54 | HEX | B#16#55 | |
| 7 | MB | 55 | HEX | B#16#66 | |
| 8 | MB | 56 | HEX | B#16#77 | |
| 9 | MB | 57 | HEX | B#16#88 | |
| 10 | | | | | |

COM_PB_1M2S_300-300(ALL)\SIMATIC 300(M)\...\S7 程序(3) RUN

b)

变量 - [VAT_2 -- @COM_PB_1M2S_300-300(ALL)\SIMATIC 300...

表格(T) 编辑(E) 插入(I) PLC(L) 变量(A) 查看(V) 选项(O) 窗口(W) 帮助(H)

| | 地址 | | 符号 | 显示格式 | 状态值 | 修改数值 |
|---|---|---|---|---|---|---|
| 1 | //S2从站接收到主站转发来的S1的数据 | | | | | |
| 2 | DB1.DBB | 0 | | HEX | B#16#11 | |
| 3 | DB1.DBB | 1 | | HEX | B#16#22 | |
| 4 | DB1.DBB | 2 | | HEX | B#16#33 | |
| 5 | DB1.DBB | 3 | | HEX | B#16#44 | |
| 6 | DB1.DBB | 4 | | HEX | B#16#55 | |
| 7 | DB1.DBB | 5 | | HEX | B#16#66 | |
| 8 | DB1.DBB | 6 | | HEX | B#16#77 | |
| 9 | DB1.DBB | 7 | | HEX | B#16#88 | |
| 10 | | | | | | |

COM_PB_1M2S_300-300(ALL)\SIMATIC 300(S2)\...\S7 程序(4 RUN

c)

图 9-34

a）主、从站打包通信的监控 b）主、从站打包通信的监控结果
c）主、从站打包通信的监控结果

## 9.4 工业以太网通信

### 9.4.1 简介

现场总线的出现，对于实现面向设备的自动化系统起到了巨大的推动作用，但现场总线这类专用实时通信网络具有成本高、速度低和支持应用有限等缺陷，再加上总线通信协议的多样性，使得不同总线产品不能互连、互用和互操作等，因而现场总线工业网络的进一步发展受到了极大的限制。

随着以太网技术的高速发展及它的 80% 的市场占有率和现场总线的明显缺陷，特别是高速以太网的出现使得以太网能够克服自己本身的缺陷，进入工业领域成为工业以太网，因而使得人们可以用以太网设备去代替昂贵的工业网络设备。工业以太网提供了针对制造业控制网络的数据传输的以太网标准。该技术基于工业标准，利用了交换以太网结构，有很高的网络安全性、可操作性和实效性。

工业以太网以其特有的低成本、高时效、高扩展性和高智能的魅力，吸引着越来越多的制造业厂商。工控领域的各大厂商纷纷研发出适合自己工控产品且兼容性强的工业以太网，其中应用最为广泛的工业以太网之一是德国西门子公司研发的 SIMATIC NET 工业以太网。它提供了开放的、适用于工业环境下各种控制级别的通信系统，这些通信系统均基于国家和国际标准，符合 ISO/OSI 网络参考模型。SIMATIC NET 工业以太网主要体系结构是由网络硬件、网络部件、拓扑结构、通行处理器和 SIMATIC NET 软件等部分组成。

（1）SIMATIC NET 工业以太网基本类型和网络硬件

SIMATIC NET 工业以太网有两种类型，分别为 10 Mbit/s 工业以太网和 100 Mbit/s 工业以太网。它是利用带传输技术，基于 IEEE802.3 利用 CSMA/CD 介质访问方法的单元级和控制级传输网络。在西门子工业以太网中，通常使用的物理传输介质是屏蔽双绞线（TP）、工业屏蔽双绞线（ITP）和光纤。

（2）SIMATIC NET 工业以太网网络部件

SIMATIC NET 工业以太网网络部件包括工业以太网链路模板、工业以太网交换机和工业以太网链路模块。

（3）SIMATIC NET 工业以太网的拓扑结构

SIMATIC NET 工业以太网的拓扑结构包括总线型拓扑结构、环形拓扑结构和环网冗余。

（4）SIMATIC NET 工业以太网通信处理器

常用的 SIMATIC NET 工业以太网通信处理器（CP）包括用在 S7 PLC 站上的处理器 CP243－1 系列、CP343－1 系列、CP443－1 系列以及用在 PC 上的网卡，并提供 ITP、RJ45 及 AUI 等以太网接口。它们以 10/100 Mbit/s 的速度将 PLC 或 PC 连接至工业以太网。CP 系列模板是为 S7 系列 PLC 在组成工业以太网进行通信时使用的，通过 CP 系列模板用户可以很方便地将 S7 系列 PLC 通过以太网进行连接，并且支持使用 STEP7－Micro/WIN32 软件。通过以太网对 S7 系列 PLC 进行远程组态、编程和诊断。

### 9.4.2 多台 S7－300 之间的 IE 通信

在生产现场，用户会遇到几台 S7－300 的 PLC 组成小型的局域网实现互相通信的情况，

在本章，我们将采用3台S7 315－2PN/DP，通过建立S7连接来说明多台S7－300 PLC的工业以太网的组网技术。

1．网络组建

本例中所使用的CPU是3台315－2PN/DP，通过集成的PN口连接到局域网上。与前面章节一样，首先创建一个新的项目，项目名称为“COM_ET_3300”，在项目内依次插入3个300站点：“SIMATIC 300（1）”、“SIMATIC 300（2）”、“SIMATIC 300（3）”。接下来是分别对3个站点进行组态。

（1）SIMATIC 300（1）站的硬件组态

首先做基本的硬件组态。在插入CPU315－2PN/DP时，系统提示是否组建以太网对话框，如图9-35所示。点击“属性”按钮，在图9-36内新建一个网络“Ethernet（1）”，并输入IP地址“192.169.10.60”，子网掩码“255.255.255.0”。点击“确定”按钮。双击2号插槽内的“CPU 315－2PN/DP”，在如图9-37所示CPU属性对话框的“周期/时钟存储器”内勾选“时钟存储器”，并输入存储器字节号为100。点击“确定”按钮。




图9-35　PN-IO的属性配置




图9-36　输入服务器的IP地址

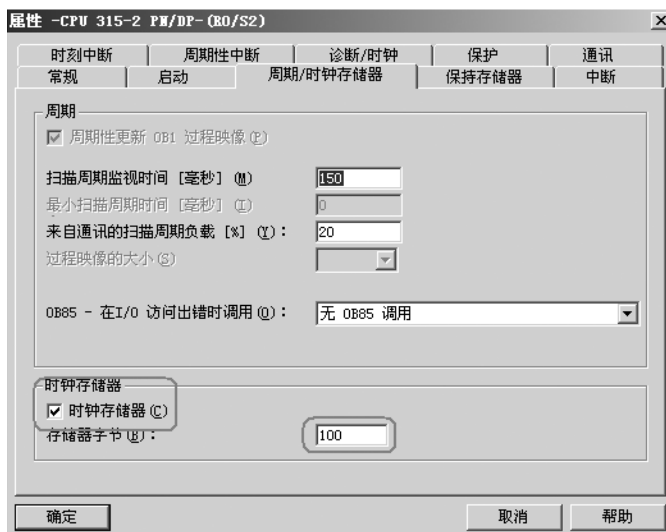



图 9-37 时钟存储器配置

在硬件组态管理器界面下，对刚才的组态进行编译保存。

（2）SIMATIC 300（2）站的硬件组态

SIMATIC 300（2）站的硬件组态的步骤和内容与 SIMATIC 300（1）站的组态一样，只不过该站的 IP 地址改为“192.168.10.61”，子网掩码依然是“255.255.255.0”；同时也设定 MB100 为时钟存储器，进行编译保存。

（3）SIMATIC 300（3）站的硬件组态

SIMATIC 300（3）站的硬件组态也是同第一站和第二站，但是该站的 IP 地址改为“192.168.10.62”，子网掩码依然是“255.255.255.0”；同时也设定 MB100 为时钟存储器。编译保存。在“SIMATIC Manager”下打开“组态网络”对话框，如图 9-38 所示。

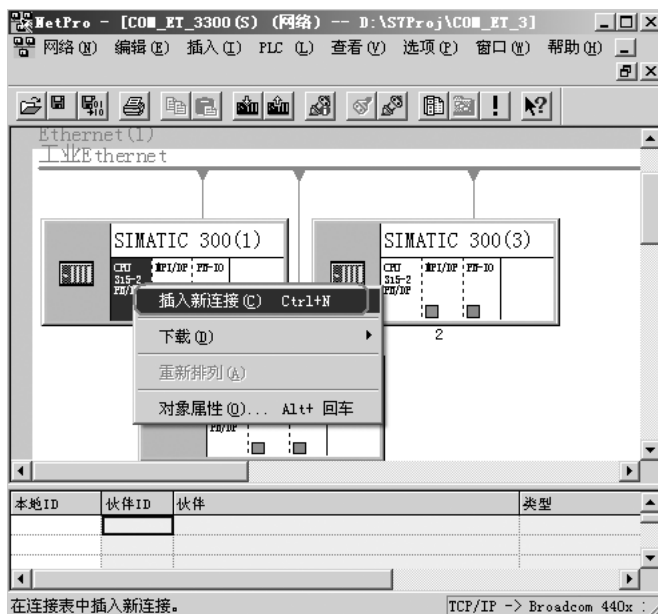



图 9-38 在“组态网络”里插入新连接

（4）建立 S7 连接

在如图 9-33 所示“组态网络”下选择“SMATIC 300（1）”站的“CPU 315 -2PN/DP”，右击并选择“插入新连接”，出现如图 9-39 所示的“插入新连接”对话框。

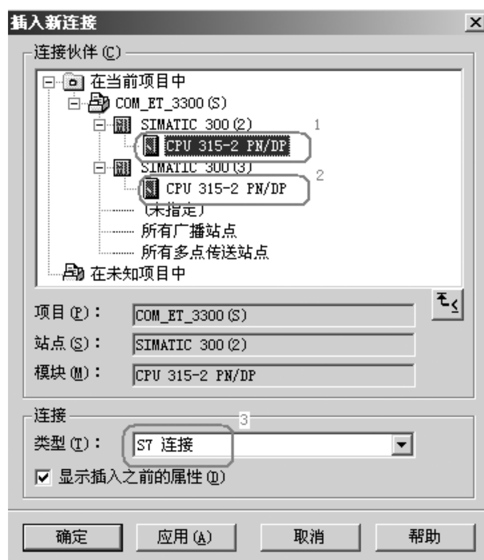



图 9-39　插入新连接

1、2 号框显示的是能够与“SMATIC 300（1）”站建立连接的站点，用户可以选择其中的一个。3 号框内是可以建立的连接类型。通过了解 315 -2PN/DP 的 CPU 属性（在硬件组态内双击 CPU 即可）可知：单独由该 CPU 可以建立 S7 连接、MPI、PROFIBUS 通信，或者作为 PROFINET IO 的控制器。所以在本例中的连接类型只能选择“S7 连接”，其他的连接如 TCP、TCP-ON-ISO、ISO 等连接需要能够支持的 CP 接口或模块。

在这里选择 1 号框的“SMATIC 300（2）”站，并选择“S7 连接”。点击“确定”按钮。在随后出现的对话框内选择“建立激活的连接”，另外要注意的是“本地 ID”号采用默认；用户要特别注意 ID 号，在后续的通信程序中要用到。点击“确定”按钮。这样就在“SMATIC 300（1）”站与“SMATIC 300（2）”站之间建立了一个 S7 类型的连接，使用 ID 号来标记这条连接就是“1 -1”连接通道。这条连接在“SMATIC 300（1）”站来看是 1 号连接，在“SMATIC 300（2）”站来看也是 1 号连接，这是因为在图 9-40 内的 ID 号，我们使用的都是默认值。为了不产生混乱，以后都采用系统默认值。

在图 9-38 内选择“SMATIC 300（2）”站的“CPU 315 -2PN/DP”，同样的步骤插入一个新连接。在图 9-39 内选择“SMATIC 300（3）”站的“CPU 315 -2PN/DP”，建立 S7 连接。在图 9-40 内选择“建立激活连接”，ID 号取默认值。点击“确定”按钮。这样在“SMATIC 300（2）”站和“SMATIC 300（3）”站之间建立一个“2 -1”连接。在“SMATIC 300（2）”站内这条连接的 ID 号为 2，表示该连接在该站内是第 2 条连接；而在“SMATIC 300（3）”站内这条连接的 ID 号为 1，表示该连接在该站内是第 1 条连接。

选择“SMATIC 300（3）”站的“CPU 315 -2PN/DP”，与“SMATIC 300（1）”站之间插入一个新连接，ID 号依然采用默认值，则该连接号为“2 -2”连接；在“SMATIC 300

(1)”站内这条连接的 ID 号为 2，表示这是该站内的第 2 条连接；而在“SMATIC 300（3）”站内这条连接的 ID 号也为 2，表示该连接是该站内的第 2 条连接。




图 9-40　S7 连接的属性

图 9-41 显示了 SMATIC 300（1）站内建立的所有连接情况，包括 ID 号、通信双方站点，连接类型等。用户可以双击某个连接以便修改该连接的参数。

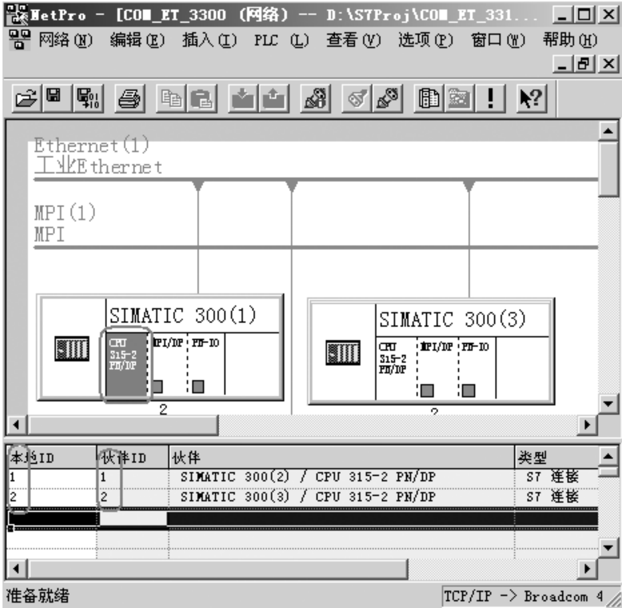



图 9-41　SMATIC 300（1）站的连接

点击图 9-41 中其他站的“CPU 315 -2PN/DP”，在下方同样显示出该站所建立的连接情况。

至此，我们在网络内建立了 3 条连接，它们之间的关系如表 9-5 所示。

**表 9-5 连接与 ID 号的关系**

| 站号 | SMATIC 300（1） | SMATIC 300（2） | SMATIC 300（3） |
|---|---|---|---|
| ID 号 | 1 | 1 | |
| 连接 | ←—— | ——→ | |
| ID 号 | | 2 | 1 |
| 连接 | | ←—— | ——→ |
| ID 号 | 2 | | 2 |
| 连接 | ←—— | ———— | ——→ |

组态完网络后，接下来应该对上面的组态进行编译保存。在图 9-41 内点击“编译保存”；如果编译结果有错误，则根据报错信息，找出错误所在并改正，如果编译无错误，则等待前面的硬件组态下载后，也下载到 PLC 内。

**2. 程序编写**

本例中所使用的 PLC 属 S7－300 系列，在进行基于 TCP/IP 通信时，需要调用库内“Standard Library”下“Communication Blocks”内的 FB12“BSEND”和 FB13“BRCV”或者 FB8“USEND”、FB9“URCV”。

FB12“BSEND”用来向类型为“BRCV”的远程伙伴 FB 发送数据。通过这种类型的数据传送，可以在通信伙伴之间为所组态的 S7 连接传输更多的数据，即可以为 S7－300（扩展 CP 模块）发送多达 32768 个字节，为 S7－400 发送多达 65534 个字节，以及通过集成接口为 S7－300 发送多达 65534 个字节的数据。另外，要发送的数据区是分段的。各个分段单独发送给通信伙伴。通信伙伴在接收到最后一个分段时对此分段进行确认。

FB8“USEND”向类型为“URCV”的远程伙伴 FB 发送数据。执行发送过程而不需要和伙伴进行协调。也就是说，在进行数据传送时不需要伙伴 FB 进行确认。

FB 13“BRCV”接收来自类型为“BSEND”的远程伙伴 SFB/FB 的数据。在收到每个数据段后，向伙伴 SFB/FB 发送一个确认帧，同时更新 LEN 参数。

为了获得大数据量的、稳定的通信，该小节采用 FB12 和 FB13 进行双边编程（在通信双方都调用发送/接收程序）。当然用户也可以采用 FB14“GET”、FB15“PUT”实现单边编程。关于单边编程，本章不再叙述其编程过程和通信结果。

我们先来认识 FB12“BSEND”和 FB13“BRCV”模块的外形结构以及引脚功能。它们的结构如图 9-42 所示，引脚功能参见表 9-6 和表 9-7。

**表 9-6 FB12 的引脚说明**

| 引　脚 | 功 能 说 明 |
|---|---|
| EN | 模块的使能端，为 1 时模块准备发送 |
| REQ | 上升沿触发数据的发送 |
| R | 上升沿中止数据的发送 |
| ID | 连接号，可参见表 9-3 中所示 WORD 型数据 |
| R_ID | 标记本次发送的数据包号 DWORD 型数据 |

（续）

| 引　脚 | 功能说明 |
| --- | --- |
| SD_1 | 发送数据存储区，可以使用指针 |
| LEN | 数据发送的长度 |
| NDR | 作业起动与否的标志 |
| ERROR | 与 STATUS 配合使用，通信的报错状态 |
| STATUS | 用数字表示通信错误的类型 |

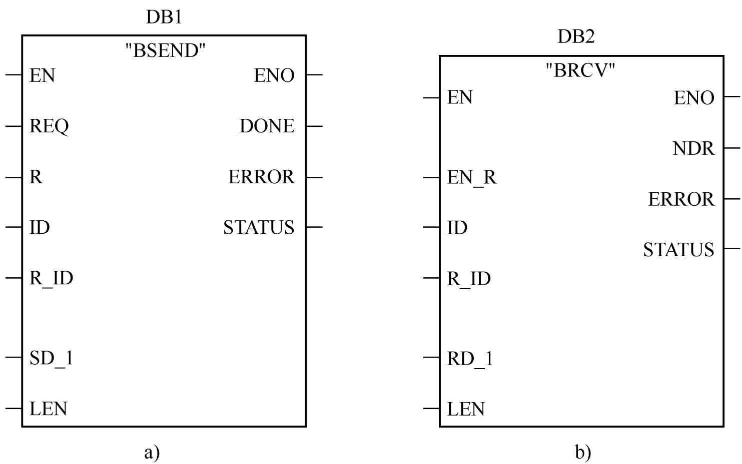



图 9-42　FB12 与 FB13 的结构图

**表 9-7　FB13 的引脚说明**

| 引　脚 | 功能说明 |
| --- | --- |
| EN | 模块的使能端，为 1 时模块才能接收 |
| REQ | 高电平准备接收数据 |
| ID | 连接号，必须与发送端对应为同一连接 |
| R_ID | 标记本次接收的数据包号，必须与发送端相同 |
| RD_1 | 接收的数据存储区，可以使用指针 |
| LEN | 数据接收的长度 |
| DONE | 数据发送作业的状态，1 发送完，0 未发送完 |
| ERROR | 与 STATUS 配合使用，通信的报错状态 |
| STATUS | 用数字表示通信错误的类型 |

下面来编写 3 个站的通信程序，该小节所采用的测试思路是：在“SMATIC 300（1）”站内发送一个数据“DW#16#33333”给“SMATIC 300（2）”站，“SMATIC 300（2）”站接收到该数据后再发送给“SMATIC 300（3）”站，“SMATIC 300（3）”站接收到该数据后再转发给“SMATIC 300（1）”站，通过“SMATIC 300（1）”站内的 MD10 所接收的由“SMATIC 300（3）”站发来的数据验证整个网络是否能够正常通信。

下面列出“SMATIC 300（1）”和“SMATIC 300（2）”的通信程序、“SMATIC 300（2）”和“SMATIC 300（3）”，以及“SMATIC 300（3）”和“SMATIC 300（1）”的通信程序。

其中图 9-43 中“REQ”引脚连接的是 M100.7，这是在硬件组态的时候所做的时钟存储器，它可以发出频率为 0.5Hz 的脉冲序列，在脉冲的上升沿触发数据发送作业。而在图 9-44 中“EN_R”端使用了 M50.0 的常闭触点，使得 FB13 时钟处于准备接收状态。

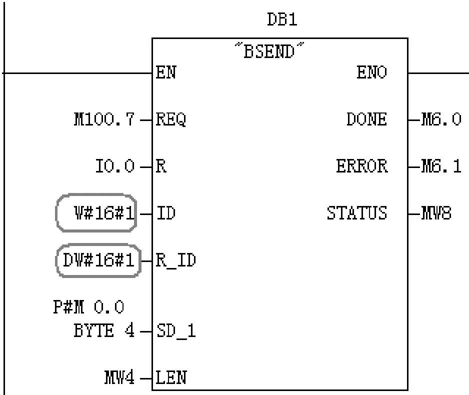


图 9-43 “SMATIC 300（1）”站的数据发送

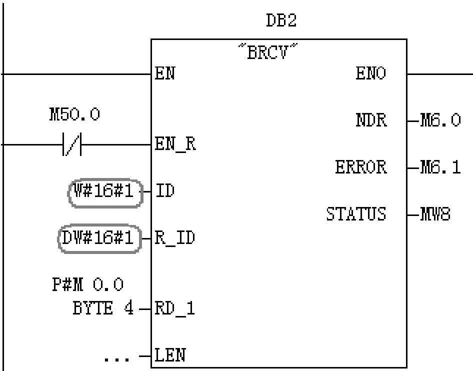


图 9-44 “SMATIC 300（2）”站的数据接收

另外，“R_ID”在通信双方必须相同，否则是不能通信，这个值可以由用户自己设定。

ID 号在通信双方可能是相同的，也可能不相同的（如表 9-5 所示），取决于通信时采用的是哪一条连接，一旦连接通道确定下来，则编程的时候双方的 ID 号就已经定下来了。例如表 9-5 中“SMATIC 300（2）”和“SMATIC 300（3）”的通信连接，在“SMATIC 300（2）”方 ID 就是 2，而在“SMATIC 300（3）”方的 ID 号就是 1。这里要特别注意。

“SMATIC 300（2）”和“SMATIC 300（3）”的通信以及“SMATIC 300（3）”和“SMATIC 300（1）”的通信程序不再一一列出，相关程序可以参见所附光盘中本小节的项目。

**3. 项目下载及监控**

（1）下载硬件组态

可以通过 MPI 或者 TCP/IP 协议采用 3.6 小节所述步骤下载硬件组态。采用 TCP/IP 下载的时候，用户最好保证编程 PC 始终和目的站的 IP 地址处于同一个子网内（不论是下载前还是下载后），否则会产生“无法和对方建立连接”的问题。

（2）下载网络组态

对图 9-41 所示的网络组态进行编译保存后，必须将 3 个站点的网络连接通道情况下载到各自的站点内。在图 9-41 内选择“SMATIC 300（1）”站的“CPU 315－2PN/DP”，直接点击下载按钮即可。同样的办法将“SMATIC 300（2）”站、“SMATIC 300（3）”站的连接情况下载到各自的 PLC 内。

（3）下载整个项目

在“SIMATIC Manager”下分别选择 3 个站点，依次点击下载按钮，在随后的对话框中点击“是”按钮即可将整个项目下载到 PLC 内。

观察各个站点 PLC 的指示灯，如果有红色灯亮，检查原因，重复上述工作，直至故障

排除。

下载完项目，在各个站点的“块”内分别插入变量表，如图9-45所示，在各自的变量表内输入如下变量，打开监控按钮，修改“SMATIC 300（1）”站中MD0的值为“DW#16#33333”，可以观察到2、3、4号框内分别收到该数据。表明基于3台315－2PN/DP的以太网通信网络组态正确，通信正常。

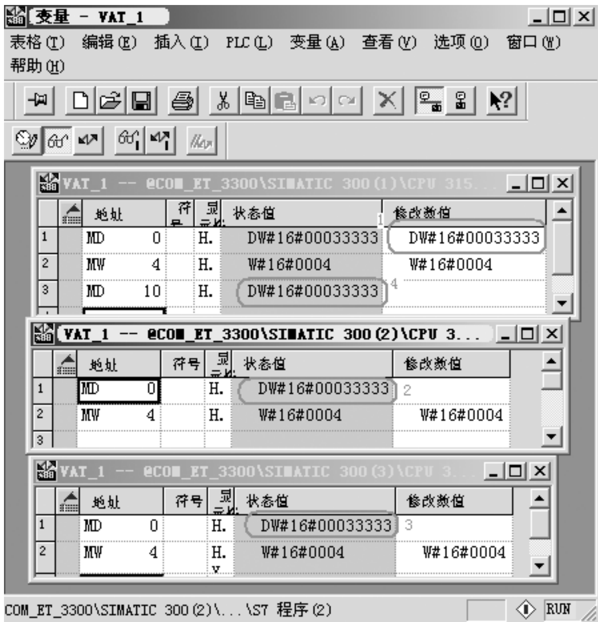


图9-45　3台PLC的以太网通信结果

# 第 10 章　西门子人机界面技术

## 10.1　人机界面简介

### 10.1.1　人机界面的基本概念

人机界面（Human-Machine Interface，HMI）是操作人员与 PLC 之间进行人机对话和相互作用的接口设备。近年来，人机界面的价格大幅下降，其应用越来越广泛，已经成为现代工业控制系统不可缺少的设备之一。

过去用按钮、开关和指示灯等作人机界面装置，它们提供的信息量少，而且操作困难，需要熟练的操作人员来操作。如用七段数字显示器来显示数字，用拨码开关来输入参数，占用的 PLC 的 I/O 点数多，硬件成本高，有时还需要自制印制电路板。

HMI 是操作员通过输入单元（如触摸屏、键盘、鼠标等）写入工作参数或输入操作命令，与控制系统之间进行对话和相互作用的数字设备。人机界面装置是操作人员与 PLC 之间双向沟通的桥梁，很多工业被控对象要求控制系统具有很强的人机界面功能，用来实现操作人员与计算机控制系统之间的对话和相互作用。人机界面装置用来显示 PLC 的 I/O 状态和各种系统信息，接收操作人员发出的各种命令和设置的参数，并将它们传送到 PLC。人机界面装置一般安装在控制屏上，能够适应恶劣的现场环境，可靠性好。在环境条件较好的控制室内可以用计算机作为人机界面装置。

### 10.1.2　人机界面的分类

人机界面由硬件和软件共同组成。

① HMI 硬件：一般分为运行组态软件程序的工控机（或 PC）和触摸屏两大类。

② HMI 软件：运行于 PC Windows 操作系统下的组态软件，例如德国西门子公司的组态软件 WinCC，美国 Wonderware 公司的组态软件 InTouch；运行于西门子触摸屏中的组态软件 WinCC Flexible。

### 10.1.3　人机界面的功能

人机界面最基本的功能是显示现场设备（通常是 PLC）中开关量的状态和寄存器中数字变量的值，用监控画面向 PLC 发出命令，并修改 PLC 寄存器中的参数。另外还包括如下功能：

① 过程可视化（设备工作状态显示，如指示灯、按钮、文字、图形、曲线等）。

② 操作员对过程的控制（数据、文字输入操作，打印输出）。

③ 显示报警。

④ 记录设备生产数据记录。

⑤ 脚本（简单的逻辑和数值运算）。

⑥ 应用于通信（可连接多种工业控制设备组网）频率高的场合。

如图 10-1 所示为一个具有集中功能的 HMI 监控系统，最上层是管理级的工控机，承担中心管理功能，它通过工业以太网与触摸屏相连。触摸屏处于现场级，主要完成现场控制功能，控制级的 PLC 通过 PROFIBUS 与触摸屏相连。

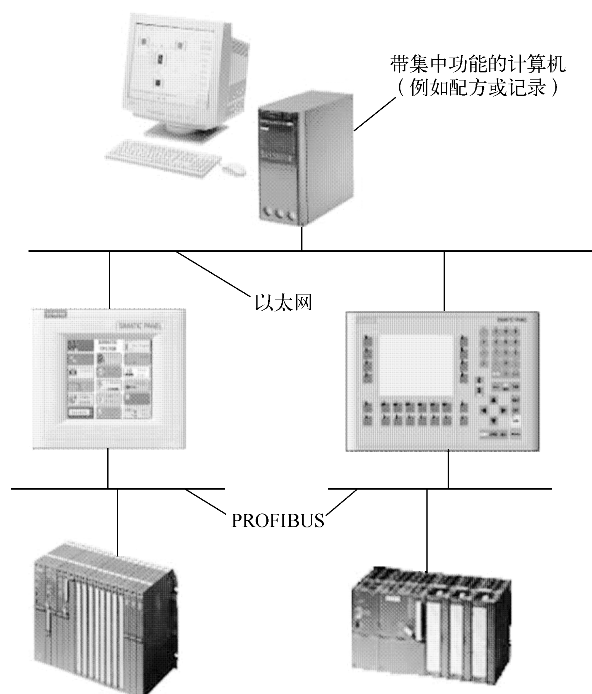



图 10-1　具有集中功能的 HMI 监控系统

如图 10-2 所示为远程访问 HMI 设备系统。在 WinCC Flexible 中使用 Sm@rtService 选件，可以通过网络（Internet、LAN）在地球的另一端从管理级工作站连接至 HMI 设备。

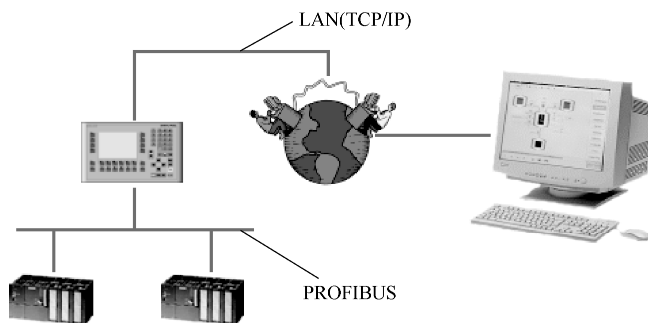



图 10-2　远程访问 HMI 设备系统

## 10.2　基于触摸屏的监控网络

### 10.2.1　触摸屏概述

#### 1. 概述

触摸屏作为一种最新的电脑输入设备，它是目前最简单、方便、自然的一种人机交

互方式。它赋予了多媒体以崭新的面貌，是极富吸引力的全新多媒体交互设备。触摸屏输入是靠触摸显示器的屏幕来输入数据的一种新颖的输入技术。它操作方式简单，使用者无需再通过键盘和鼠标，仅用手指触摸屏幕上的图形、表格或提示标志，便可从屏幕上得到其所需的诸种信息。其优点是操作简便直观、图像清晰、坚固耐用及节省空间，它可用于一切电子显示器，并可与显示器制成一体，人机交互性好、操作方便、使用灵活、效率高及输入速度快。

2. 原理和分类

触摸屏系统一般包括两个部分：触摸检测装置和触摸屏控制器。触摸检测装置安装在显示器屏幕前面，用于检测用户触摸位置，接收后送触摸屏控制器。触摸屏控制器的主要作用是从触摸点检测装置上接收触摸信息，并将它转换成触点坐标，再送给 CPU，它同时能接收 CPU 发来的命令并加以执行。

目前主要有以下几种类型的触摸屏，它们分别是电阻式（双层）、表面电容式和感应电容式、表面声波式、红外式，以及弯曲波式、有源数字转换器式和光学成像式。它们又可以分为两类，一类需要 ITO，比如前三种触摸屏，另一类的结构中不需要 ITO，比如后几种屏。目前市场上，使用 ITO 材料的电阻式触摸屏和电容式触摸屏应用最为广泛。ITO 是铟锡氧化物的英文缩写，它是一种透明的导电体。通过调整铟和锡的比例、沉积方法、氧化程度以及晶粒的大小可以调整这种物质的性能。

3. 应用

触摸屏技术方便了人们对计算机的操作使用，是一种极有发展前途的交互式输入技术。世界各国对此普遍给予重视，并投入大量的人力物力进行研发，新型触摸屏不断涌现。触摸屏在我国的应用范围非常广阔，主要是公共信息的查询，如电信局、税务局、银行、电力等部门的业务查询，城市街头的信息查询，此外应用于办公、工业控制、军事指挥、电子游戏、点歌、多媒体教学、房地产预售等。将来，触摸屏还要走入家庭。

触摸屏的发展呈现专业化、多媒体化、立体化和大屏幕化等趋势。随着信息社会的发展，人们需要获得各种各样公共信息。以触摸屏技术为交互窗口的公共信息传输系统，通过采用先进的计算机技术，运用文字、图像、音乐、解说、动画、录像等多种形式，直观、形象地把各种信息介绍给人们，给人们带来极大的方便。可以预见，随着触摸屏技术的迅速发展，触摸屏的应用领域会越来越广，性能会越来越好。

4. 使用基础

如图 10-3 所示为项目的组态阶段和运行阶段网络示意图。在项目组态阶段，通过组态计算机（PC），在组态软件 WinCC Flexible 平台上创建项目和编辑项目。WinCC Flexible 还提供了模拟运行系统，可以实现离线测试项目，即没有实际的触摸屏也可以完成功能测试。这个模拟仿真功能大大方便了开发人员。

将组态好的项目下载到触摸屏中。以 TP170A 为例，给触摸屏上电后，自动转到“Loader”（装载对话框），打开装载对话框（如图 10-4 所示）。单击“Config”（设置）按钮，打开“Transfer Settings”（传送设置对话框）（如图 10-5 所示），设置下载项目所用的协议，默认为串口传送，直接通过串行电缆下载项目，串行电缆引脚连接图如图 10-6 所示。

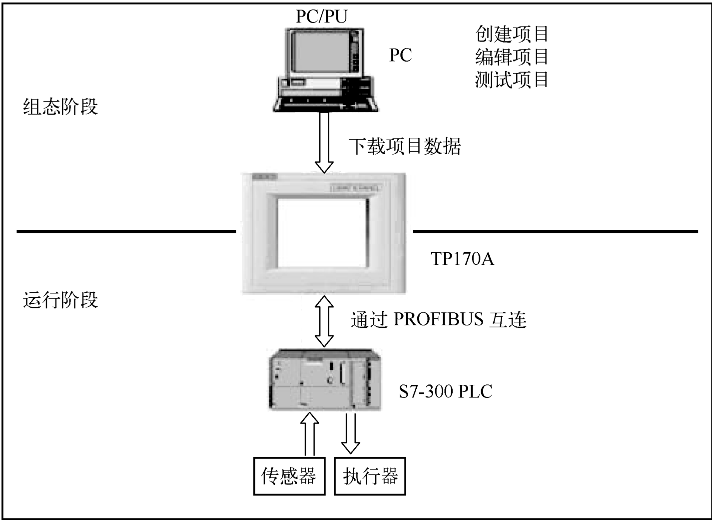



图 10-3　网络示意图

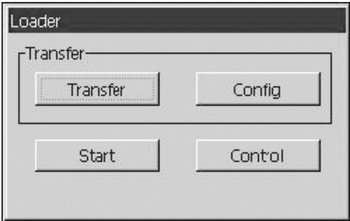



图 10-4　装载对话框

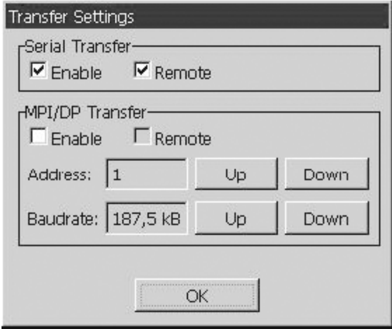



图 10-5　传送设置对话框

下载方式也可以选用“MPI/DP Transfer”（MPI/DP 传送），则必须在组态计算机中插有通信卡（例如 CP5611），传送设置对话框（如图 10-7 所示），在“Address”项中设置触摸屏的地址，一般默认为 1，传输速度一般为 187.5 kbit/s，在“Baudrate”中设置，这个速度要与组态计算机中设置的下载速度一致。单击“OK”按钮完成传送设置。

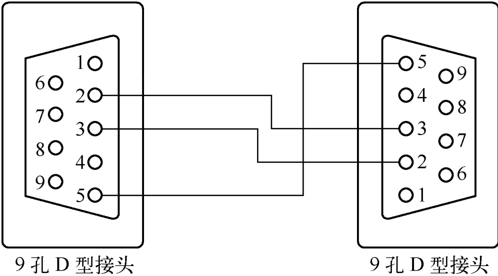



图 10-6　串行电缆引脚连接图

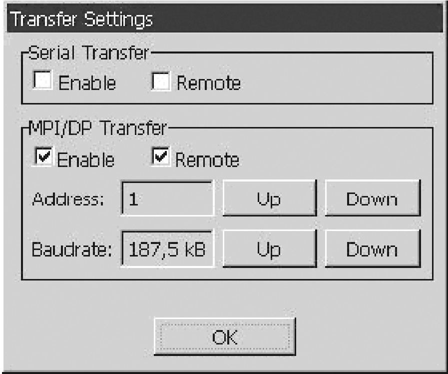



图 10-7　传送设置对话框

在项目运行阶段，触摸屏 TP170A 与 PLC 通过 RS－485 通信口连接。利用 PROFIBUS 电缆连接，实现与 PLC 通信。通过 PLC 完成数据采集和设备控制，并将系统设备的实时状态在触摸屏上显示出来。

### 10.2.2　组态软件 WinCC Flexible 基础

1．WinCC Flexible 简介

WinCC Flexible 组态软件适用于从单用户、多用户到基于网络的工厂自动化控制和监视。几乎所有的西门子触摸屏都可使用。ProTool 适用于单用户系统，逐渐被 WinCC Flexible 取代。WinCC Flexible 简单、灵活、高效、易于上手，用于面向机器和过程的应用，并能实现满足不同级别和性能的设备的操作和运行。WinCC Flexible 使自动化过程更加透明，组态更加简单，反应更加迅速，同时减少了现场的操作人员。即使在复杂的工作环境下，HMI 也能确保可靠的信息交换。

2．WinCC Flexible 的系统组成与功能

通过使用 WinCC Flexible，可以实现个性化人机界面，应用于不同的行业、不同的用户。一次创建画面模块，可以多次重复使用，同时使用各种智能工具，可以使开发组态更加容易。也可以通过使用 Visual Basic 脚本来动态显示对象，方便访问文本、图形或条形图等屏幕对象属性。WinCC Flexible 提供的单独接口，如 OPC 和用于过程控制的 OLE，可用于集成自动化解决方案到具有不同设备的工厂中或办公环境中。Sm@ rtAcess 选件提供系统对过程数据直接访问的支持。另外，还可以在中央管理控制中心集中分析归档过程值数据或从操作员站观察机器设备状态。Sm@ rtService 选件允许通过 Web 进行控制、诊断和服务，通过 Email 或文本短信发送重要的报警到维护人员，这样通过人机界面实现远程的信息交换。

HMI 系统承担下列任务：

① 过程可视化：过程显示在 HMI 设备上，HMI 设备上的画面可根据过程变化动态更新。

② 操作员对过程的控制：操作员可以通过 GUI（图形用户界面）来控制过程。例如，操作员可以预置控件的参考数值或者起动电动机。

③ 显示报警：当超出设定值时，会自动触发报警。

④ 归档过程值和报警：可以记录报警和过程值，该功能使用户记录过程顺序，并检索以前的生产数据。

⑤ 过程值和报警记录：可以输出报警和过程值报表。例如，用户可以在某一轮班结束时打印输出生产数据。

⑥ 过程和设备的参数管理：可以将过程和设备的参数存储在配方中。例如，可以一次性将这些参数从 HMI 设备下载到 PLC，以便改变产品版本进行生产。

### 10.2.3　WinCC Flexible 过程通信

WinCC Flexible 与自动化层之间的数据交换需要建立通信连接才能进行。在集成的项目中，可以创建与下列应用程序的连接。

1．WinCC Flexible

在 WinCC Flexible 软件中，可以创建新的通信连接或编辑现有的通信连接。可以使用许

多预先安装的通信驱动程序，根据连接的 PLC 类型的不同，选择合适的驱动程序。

如图 10-8 所示是通信连接示意图，与触摸屏相连的有 3 个 PLC、1 个 S7－300 和 2 个 S7－200，分别对应不同的通信驱动程序。




图 10-8　通信连接示意图

在与 S7 集成的项目中，编辑器中还包含“站”、“伙伴”和“节点”的列，以用于连接组态。创建连接时，从选择列表中选择站、伙伴和连接节点。在 STEP 7 自动接收所需的连接参数。组态完成后，必须保存项目。在 WinCC Flexible 中组态的连接将不会传送给 NetPro，只能使用 WinCC Flexible 进行编辑。

如图 10-8 所示，单击“连接_3”，打开通信参数设置界面（如图 10-9 所示），选择触摸屏 TP 270 的“IF1B 接口”，此接口遵循 RS－485 协议，可以连接 MPI 网络或 PROFIBUS 网络。具体的设置和修改通信参数可以在参数框部分完成，如图 10-9 所示，设置 IF1B 接口以及 HMI 的波特率、PLC 地址和网络类型等。




图 10-9　通信参数设置

也可以选择触摸屏 TP 270 的 ETHERNET 接口，如图 10-10 所示，选择触摸屏和 PLC 的 IP 地址。此接口用于连接工业以太网。

2. NetPro

对于较大的项目，建议使用 NetPro。NetPro 是 STEP 7 中的一个组件，用于组态和设置网络连接，支持在图形的界面上组态连接。起动 NetPro 时，将显示 STEP 7 项目中的设备和子网。NetPro 具有一个网络对象目录，可用来插入附加设备或子网。在 S7 集成的项目中，

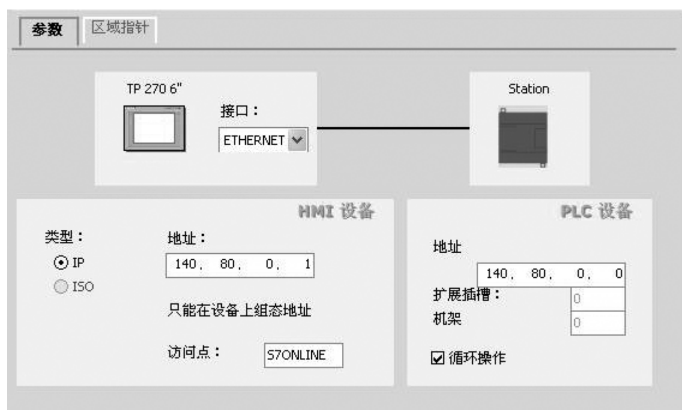



图 10-10　通信参数设置

该目录还包含 SIMATIC HMI 站对象。使用拖放操作将对象从目录插入 NetPro 的工作区域中。使用拖放操作将各个站连接至子网。使用属性对话框组态节点和子网的连接参数。然后在 NetPro 中保存组态，以便更新 WinCC Flexible 项目中的数据管理。使用 NetPro 所组态的连接将只能在 WinCC Flexible 中读取。在 WinCC Flexible 中，将只能对连接进行重新命名或输入连接的注释以及将连接设置为“在线”。对连接本身进行编辑只能专门使用 NetPro 来进行。

## 10.2.4　应用举例

本节以一个小型项目为例，创建一个包括触摸屏和 PLC 的自动化项目。进一步熟悉使用 WinCC Flexible 软件的基本方法，以及 PLC 与触摸屏的通信设置。

此项目的组态见本书附带光盘中的项目“HMI 项目 1”。

### 1. 项目要求

利用 TP170A 和西门子 S7－300 实现对某储油罐的控制，储油罐高为 100 cm，有 1 个进油阀和 1 个排油阀，油罐内部安装有液位传感器，用来采集实际油位。要求在触摸屏上实现下述控制功能：

① 在触摸屏上设置手动和自动转换开关。

② 在手动情况下，可分别控制进油阀和排油阀的开启和关闭，并可以读取储油罐实时液位值。

③ 在自动情况下，当任意设定油位高度（小于 90 cm）后，且实际油位小于设定油位时，进油阀自动打开，开始给油罐进油。当实际油位值等于或大于设定油位值时，进油阀自动关闭。

### 2. 硬件配置

① 313C－2DP。

② SM334。

### 3. 建立变量

根据项目的具体情况，设计需要的变量及意义，如表 10-1 所示，其中包括了系统的 PLC 变量和 HMI 变量。此表中只列出了系统的过程变量，即与 PLC 进行数据交换的变量，在 PLC 中均有实际的地址，其值随 PLC 程序的执行而改变。另外还有一些系统工作过程中

的间接变量并没有在此罗列。本实例中液位传感器提供的压力信号输出的电流是 4～20 mA，利用 PLC 的模拟量输入模块对其进行采集，其 I/O 地址为 PIW256。表 10-1 中的 MD4 变量是经过 PLC 程序运算后得出的实际液位值的存放地址。

表 10-1　系统变量表

| 序　　号 | PLC 变量 | HMI 变量 | 意　　义 |
|---|---|---|---|
| 1 | PIW256 | | 传感器输入值 |
| 2 | MD4 | MD4 | 实际液位值 |
| 3 | Q0.0 | M1.0 | 开阀门 1 |
| 4 | M1.1 | M1.1 | 关阀门 1 |
| 5 | Q0.1 | M1.2 | 开阀门 2 |
| 6 | M1.3 | M1.3 | 关阀门 2 |
| 7 | MD8 | MD8 | 设定液位值 |
| 8 | M1.7 | M1.7 | 手动/自动转换开关 |

### 4. 在 WinCC Flexible 中组态项目

(1) 新建变量

在“项目视图”中，双击“变量”，打开“变量编辑器”。根据表 10-1 所示创建 HMI 变量。

(2) 制作画面并连接变量

建立 2 个画面，分别命名为“欢迎画面”和“油罐监控画面”。在画面中，设置可以在 2 个画面中互相切换的按钮。我们主要的组态工作都在“油罐监控画面”中，画面设置如图 10-11 所示。

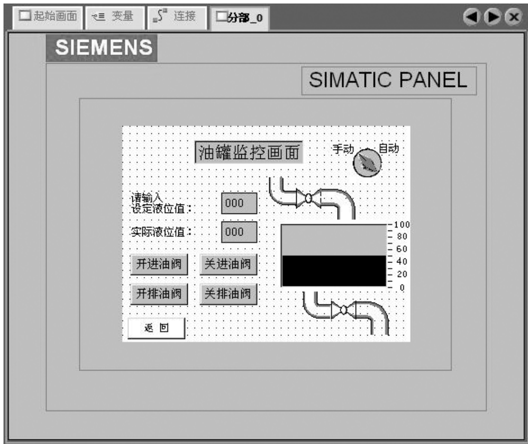



图 10-11　油罐监控画面

在“油罐监控画面”中，需要生成多个组态元素。插入 2 个 IO 域，如图 10-12 所示是其属性视图的常规项，设置模式为输入，分别与“设定液位值”变量和“实际液位值”变量相关联。插入 4 个按钮，分别命名为“开进油阀”、“关进油阀”、“开排油阀”和“关排油阀”。插入一个棒图，用于模拟显示储油罐。属性的设置比较简单，在此省略。

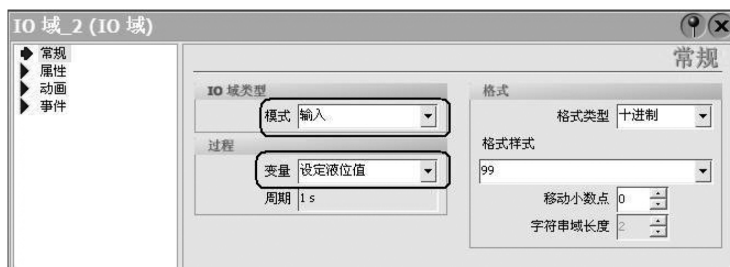



图 10-12　IO 域的属性视图的常规项

另外，画面右上角的手动/自动转换开关来自工具库，目录为 button_and_switches / Monochrom（单色）/Rotary_switches。图 10-11 中的阀门和连接管图形也来自工具库，具体目录是其图形项中的 WinCC Flexible 图像文件夹 \ Symbol Factory Graphics \ SymbolFactory 4 colors \ Pipes 和 Vavels。

（3）设置通信连接

需要设置 TP170A 与 PLC 之间的通信连接。在“项目视图”中，双击“连接”，可打开相应的画面，如图 10-13 所示。新建连接，名称为“连接_1”。在下面的“参数”选项中可以设置 HMI、网络和 PLC 设备的具体参数。设置 HMI 设备的波特率为 187500 bit/s，地址为 1，PLC 的 MPI 地址为 2（需要在 STEP 7 中做同样的设置），二者之间通过 MPI 网络互相通信。其他参数均使用默认值即可。

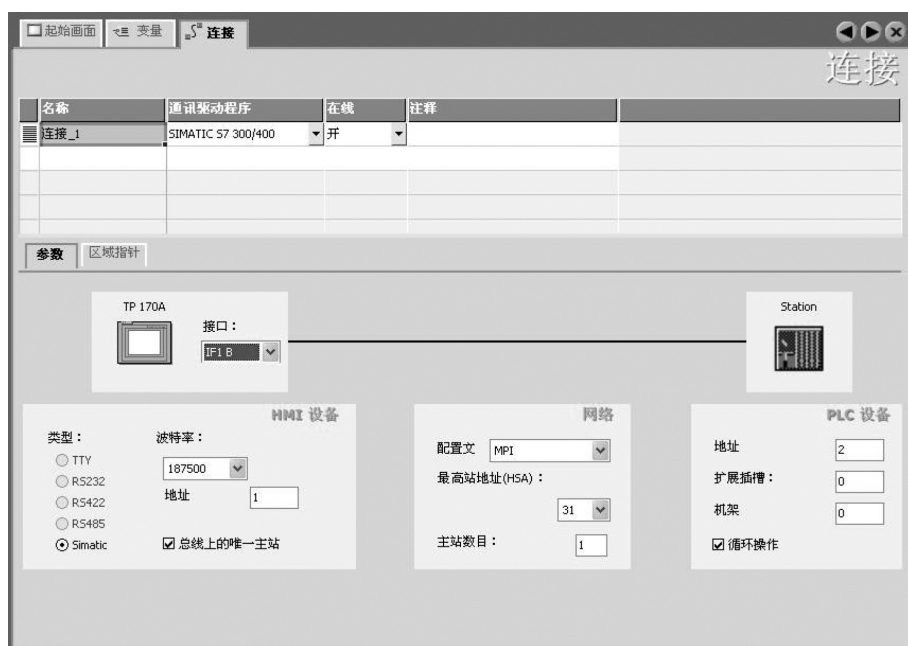



图 10-13　通信连接设置

### 5. 在 S7 -300 中编写程序

根据定义的变量，程序如图 10-14 所示。

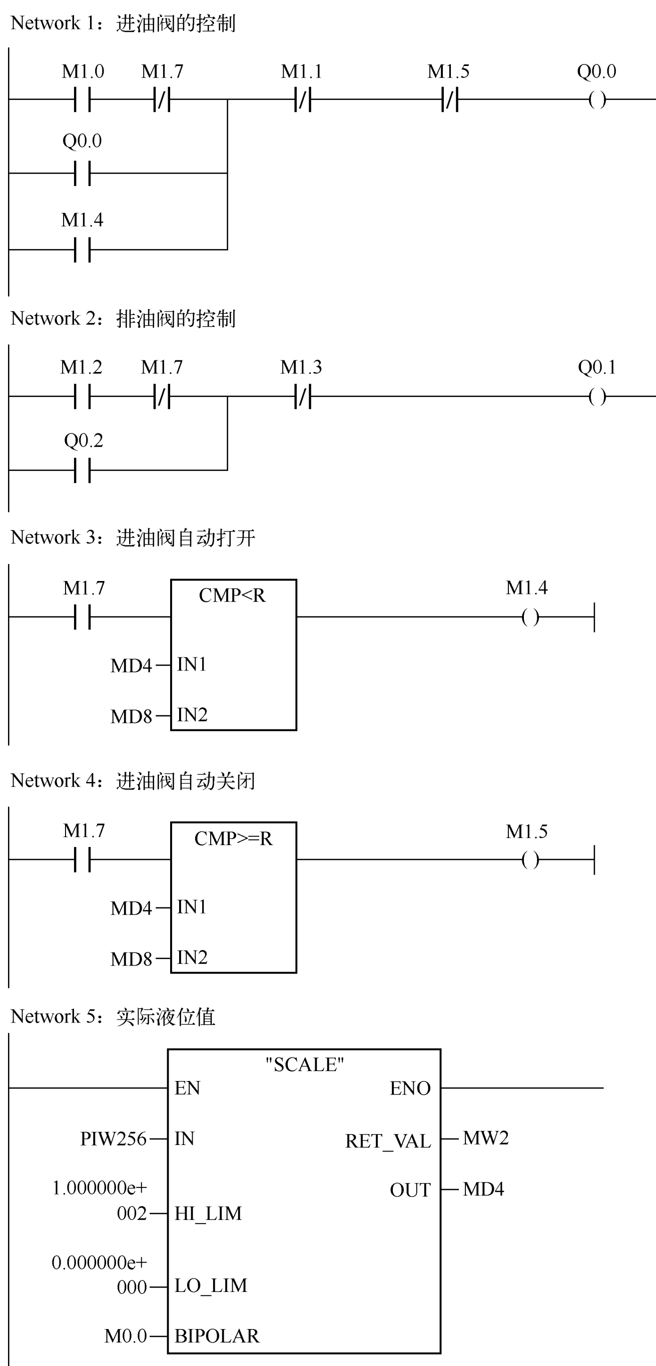



图 10-14　油罐 PLC 程序

程序模块“SCALE”是 S7－300 系统软件中自带的功能 FC105，它位于“程序元素”中的“库”，是系统已经定义好的子程序，详细目录为“Standard Library/TI－S7 Converting Blocks”，功能是完成传感器输入值的线性转换。其反作用的功能是 FC106“UNSCALE”。

在本例中，液位传感器的输出接到模拟量模块 SM334，由它完成 A/D 转换功能，是

将 4 ~ 20 mA 的标准信号转换为 0 ~ 27648 之间的数字量数据，这个数据没有实际的单位。一般来说，工程人员习惯用带有实际工程单位的工程量来计算，如 50℃的水、10 MPa 的压力等。因此，可以通过网络 5 的定标功能 FC105 来转换。假设这里的转换区间定为 0 ~ 100 cm，这样 A/D 转换的结果就介于 0 ~ 100 cm 之间了。结果存入 MD4 中，供触摸屏使用。

## 10.3 基于 PC 的工业监控网络

### 10.3.1 工控机概述

随着大规模集成电路制造技术的高度发展，PC 硬件结构做得越来越小，CPU 的运行速度越来越高，存储容量越来越大。PC 大批量生产，成本大大降低，可靠性不断提高。

目前，PC 占通用计算机 95% 以上，这是工控机的基础。工控领域的专家和技术人员自然想赋予 PC 总线更高的使命，希望让它在过程控制、制造自动化、楼宇自动化等方面扮演重要角色。工控机能在恶劣环境下（如高温、潮湿和振动等）长期可靠工作。台式工控机平均无故障时间（MTBF）为 10000 ~ 15000 h。

尤其是在我国，作为与 DCS、PLC 成鼎足之势的工业 PC 市场在逐渐扩大。各大 PLC 制造厂商，如 Siemens 公司、Rockwell Automation 公司、GE FANUC 公司、三菱电机公司均已推出自己品牌的工业 PC 产品。

### 10.3.2 组态软件 WinCC 基础

1. WinCC 简介

WinCC 是 Windows Control Center 的简称，是西门子和微软合作开发的监控系统软件。WinCC 是目前最常用的三大 SCADA（Supervisory Control And Data Acquisition，监视控制与数据采集）系统之一。目前，世界上公认的三大 SCADA 系统是 WinCC、iFix 和 InTouch。

与所有 SCADA 系统一样，WinCC 是以计算机为基础的生产过程与调度自动化系统。它可以对现场的运行设备进行监视和控制，以实现数据采集、设备控制、测量、参数调节以及各类信号报警等功能。

SIMATIC WinCC 是第一个使用最新的 32 位技术的过程监视系统，具有良好的开放性和灵活性。WinCC 是 SCADA 系统最新水平的体现，反映了 SCADA 技术发展趋势。另一方面，WinCC 技术上的继承性，可以最大程度地保护用户的投资利益。目前 WinCC 已发展成为欧洲监控技术的领导者，甚至成为业界遵循的标准。使用 WinCC 可以最大程度地提高工厂的可用性和生产效率。

现场总线系统是工厂的底层控制网络，它可以把现场设备的运行参数、状态以及故障信息等发送给上层监控系统。

2. WinCC 的系统组成与功能

WinCC 系统由资源管理器、编程接口、图形系统、消息系统、归档系统、报表系统、脚本处理系统、过程通信系统和 Windows 标准接口等部分组成，如图 10-15 所示。所有部

分协调工作，构成一个大系统。

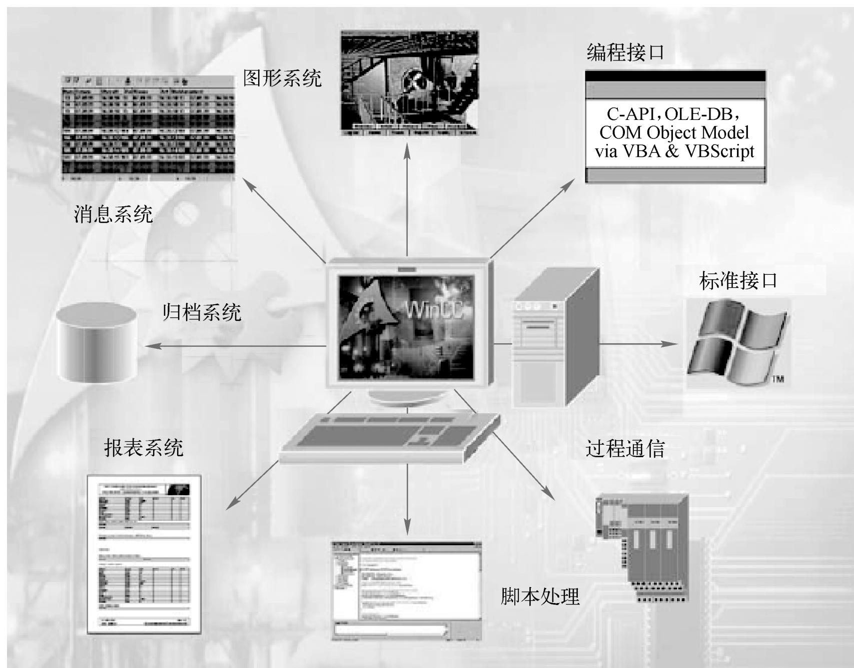



图 10-15　WinCC 的系统组成与功能

WinCC 的图形编辑器是面向对象的，并且具有友好的界面。WinCC 还内置了内容广泛全面的程序库。WinCC 采用了组态的一些新技术，包括模块化技术、在线快速组态、用于海量数据的组态工具以及提高访问透明性的交叉索引列表等。完善的组态功能显著地降低了工程和培训所需要的时间与精力，提高了效率。

WinCC 中集成了 SCADA 系统的所有功能。强大的 SCADA 系统功能是系统的基本特性。在 WinCC 中，采用全图形化的方式显示过程和状态，用来生成报表和确认事件、归档测量值和消息、记录过程和归档数据以及管理用户的访问。系统可连续记录与质量有关的顺序和事件，使系统能够始终如一地确保质量。

### 3. WinCC 的应用

基于 PC 的操作员控制和监视系统目前正处于快速发展阶段。WinCC 集生产和过程自动化于一体，实现了两者之间的整合。正是由于 WinCC 的上述特点，WinCC 在不同工业领域中得到了广泛的应用。WinCC 的主要应用领域有：汽车生产和零配件供应；化学和制药工业；能源供应和分配；塑料和橡胶工业；冶金工业；食品、饮料和烟草工业；钢铁工业；水处理和污水净化；运输行业等。

## 10.3.3　WinCC 过程通信

### 1. 数据管理器

WinCC 数据管理器管理整个项目的数据。这个数据管理器不为用户所见。它处理 WinCC 项目运行过程中产生的所有数据和存储在项目数据库中的数据。

在运行期间，它管理 WinCC 变量。WinCC 的所有应用程序必须以 WinCC 变量的形式从数据管理器中访问数据。这些应用程序包括图形运行系统、报警记录运行系统和变量记录运行系统等。

### 2. 通信驱动程序

为了使 WinCC 与各种不同类型的 PLC 进行通信，采用通信驱动程序。WinCC 通信驱动程序连接数据管理器和 PLC。通信驱动程序包括 C ++ DLL，它与数据管理器的接口进行通信。通信程序为 WinCC 变量提供过程值。

WinCC 提供了所有最重要的通信通道，用于连接到 SIMATIC S5/S7/505 控制器（如通过 S7 协议集）的通信，以及如 PROFIBUS-DP/FMS、DDE（动态数据交换）和 OPC（用于过程控制的 OLE）等非专用通道，亦能以附加件的形式获得其他通信通道。由于很多控制器制造商都为其硬件提供了相应的 OPC 服务器，因而事实上很多时候可以不受限制地将各种硬件连接到 WinCC。

与 WinCC 进行通信的实际 PLC 数量不仅取决于通信系统本身，而且取决于通信驱动程序、使用的通信卡和 PLC 的类型等。如表 10-2 所示为不同通信驱动程序下所带的 PLC 数量。

表 10-2　不同通信驱动程序下所带的 PLC 数量

| 通信类型 | 通信驱动程序 | PLC | 数量 |
|---|---|---|---|
| 串行 | S5 SERIAL 3964 | SIMATIC S5 | 2 |
| MPI | S7 MPI | SIMATIC S7 | 29 |
| PROFIBUS | S5 PROFIBUS | SIMATIC S5 | 24 |
| PROFIBUS | PROFIBUS FMS | SIMATIC S5<br>SIMATIC S7 | 32 |
| PROFIBUS | S7 PROFIBUS | SIMATIC S7－400 | 32 |
| PROFIBUS | S7 PROFIBUS | SIMATIC S7－300 | 118 |
| PROFIBUS | PROFIBUS DP | PROFIBUS DP 从站 | 126 |
| 工业以太网 | S5 以太网 TF | SIMATIC S5 | 30 |
| 工业以太网 | S5 以太网第 4 层 | SIMATIC S5 | 60 |
| 工业以太网 | S7 工业以太网 | SIMATIC S7 | 60 |

### 3. 通信结构

WinCC 数据管理器管理运行时的 WinCC 变量。各种 WinCC 应用程序从数据管理器中访问变量值。

如图 10-16 所示为 WinCC 的通信结构。数据管理器的任务是从过程中取出请求的变量值。它通过集成在 WinCC 项目中的通信驱动程序完成该过程。通信驱动程序利用其通道单元构成 WinCC 和过程处理之间的接口。在大多数情况下，基于硬件的连接利用通信处理卡实现。WinCC 驱动程序使用通信处理卡向 PLC 发送请求消息。然后，通信处理卡将相应回答消息中的请求的过程值发回 WinCC。

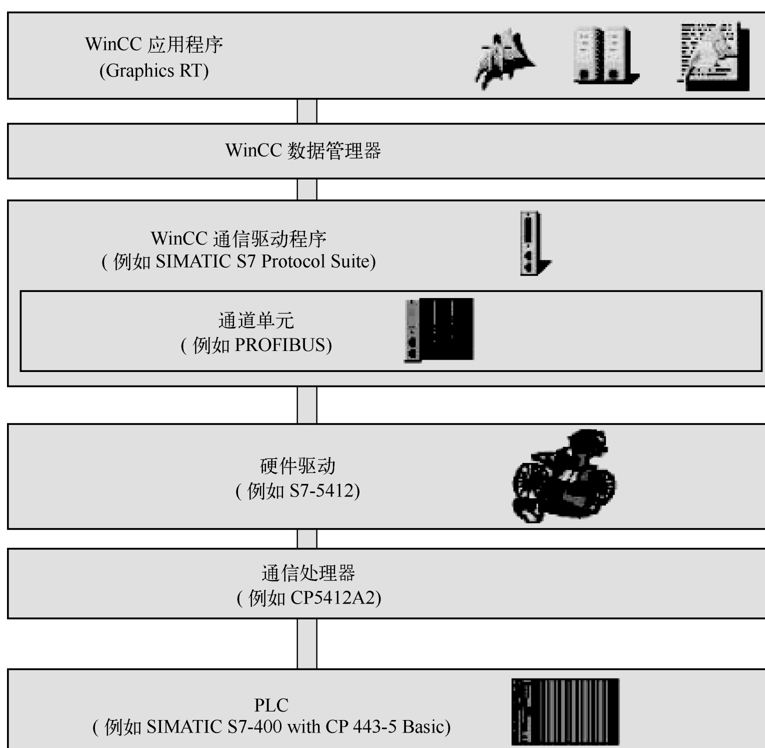



图 10-16　WinCC 通信结构图

### 10.3.4　WinCC 通信组态

下面说明在 WinCC 中建立与 PLC 的通信连接所必需的组态步骤。

**1. 添加通信驱动程序**

WinCC 中的通信通过使用各种通信驱动程序来完成。对于不同总线系统上不同 PLC 的连接，有许多通信驱动程序可用。

将通信驱动程序添加到 WinCC 资源管理器内的 WinCC 项目中。在此处，将通信驱动程序添加到变量管理器中。如图 10-17 所示，在资源管理器中，右键单击“TagManagement”，从弹出式菜单中选择“Add New Driver...”来完成该添加过程。每个通信驱动程序只能被添加到 WinCC 项目一次。

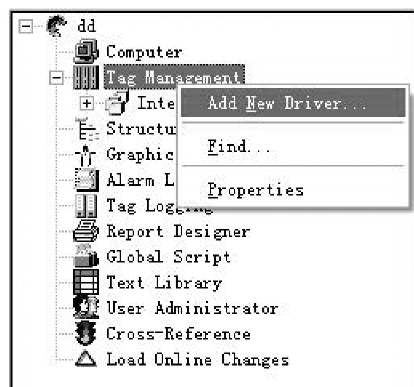



图 10-17　添加通信驱动程序

通信驱动程序是具有“.chn”扩展名的文件。计算机上安装的通信驱动程序位于 WinCC 安装文件夹的“Bin”子文件夹内。

将通信驱动程序添加到 WinCC 项目中后，它就会在“WinCC 资源管理器”中列出，在“变量管理器”下作为与“内部变量”相邻的子条目。

**2. 通道单元**

变量管理器中的通信驱动程序条目至少包含一个子条目，这就是通常所说的通信驱动程序的通道单

元。每个通道单元构成至少一个确定的从属于硬件驱动程序和 PC 通信模块的接口。必须定义由通道单元寻址的通信模块。

在系统参数对话框中分配该通信模块。通过单击鼠标右键相应的通道单元条目，并从弹出式菜单中选择系统参数来打开“System Parameter”，如图 10-18 所示。

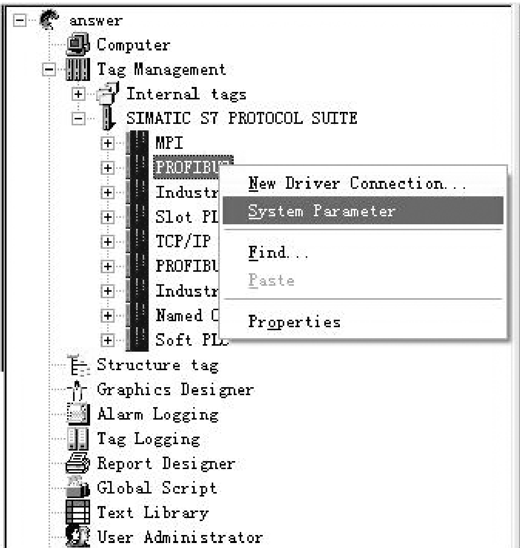



图 10-18　添加系统参数示意图

该对话框的外观取决于所选择的通信驱动程序。通常，在此处指定通道单元使用的模块。另外，也可能需要指定附加的通信参数。如图 10-19 所示为 PROFIBUS 的连接参数设置对话框。

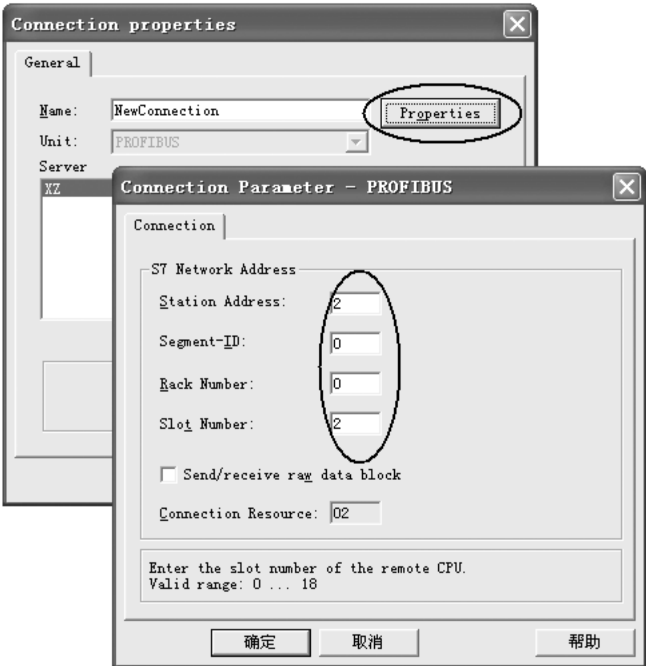



图 10-19　PROFIBUS 的连接参数设置

### 3. 连接

通道单元要读写 PLC 的过程值，必须建立与该 PLC 的连接。通过单击鼠标右键相应的通道单元条目，并从弹出式菜单中选择“新驱动程序的连接”来建立新的连接，如图 10-20 所示。要设置的连接参数取决于所选择的驱动程序。必须为连接分配一个在该项目中惟一的名称。附加的参数通常指定可到达的通信伙伴。




图 10-20　新建 PROFIBUS 协议的驱动连接

### 4. WinCC 变量

要获得 PLC 中的某个数据，必须组态 WinCC 变量。相对于没有进行驱动程序连接的内部变量，它们也被称为外部变量。要创建 WinCC 变量，可以单击鼠标右键相应的连接条目，并从弹出式菜单中选择“New Tag...”。打开变量属性对话框，如图 10-21 所示，可以定义变量的名称、地址和数据类型等属性。

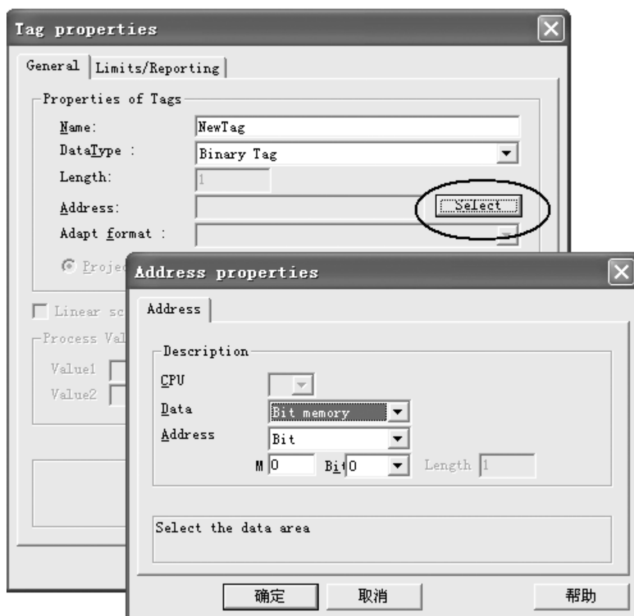



图 10-21　变量属性对话框

### 5. 通信检测

检测 WinCC 和 PLC 之间的通信是否正常。在“Set PG/PC interface”对话框中，设置为“S7ONLINE（STEP 7）CP5611（PROFIBUS）”，单击“Diagnostics”按钮，打开诊断对话框，如图 10-22 所示；单击“TEST”和“READ”按钮，若在图中“Bus Nodes”相应区域中出现“√”，则表示相应的站点通信正常。

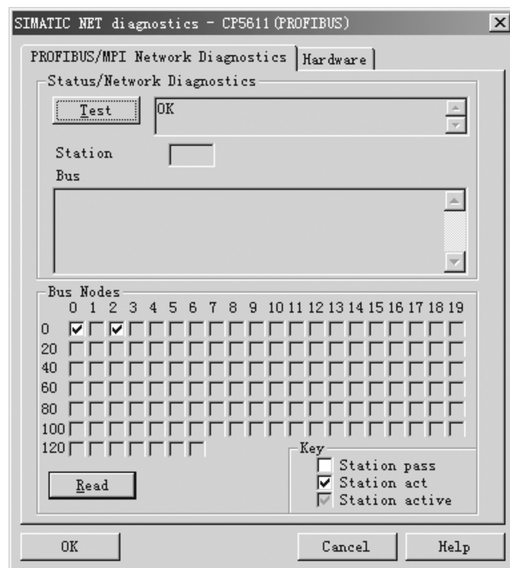



图 10-22　通信检测

# 第 11 章　PLC 在实际工程中的应用

学习 PLC 的最终目的是把它应用到实际的工程控制系统中去。在前面几章中，对 PLC 的基本配置、指令系统和编程方法有了一定的掌握之后，就可以利用 PLC 构成一个实际的控制系统，这种系统的设计就是 PLC 的应用设计。前面章节涉及的一些实例都是我们通常在实验室中的一些应用，而由 PLC 所构成的工业中的实际系统，与实验室中的应用是有所区别的。本章我们将引用实际工程应用阐述系统的设计。

## 11.1　PLC 控制系统的设计

### 11.1.1　设计原则

每一个成熟的 PLC 控制系统在设计时要达到的目的都是实现对被控对象的预定控制。为实现这一目的，在进行 PLC 控制系统的设计时，应遵循以下的基本原则：

（1）最大限度地满足被控对象的控制要求

系统设计前，除了了解被控对象的各种技术要求外，还应深入现场进行调查研究，搜集资料，并与工艺师和实际操作人员密切配合，共同拟定电气控制方案。

（2）系统结构力求简单

在满足控制要求的前提下，力求使控制系统简单、经济、操作及维护方便。对一些过去较为繁琐的控制可利用 PLC 的特点加以简化，通过内部程序简化外部接线及操作方式。

（3）保证控制系统的安全、可靠

控制系统的安全性、可靠性是提高生产效率和产品质量的必要保证，是衡量控制系统优劣的因素之一。为确保系统的安全性、可靠性可适当增加外部安全措施，如急停电源等，进一步保证系统的安全，同时采取“软硬兼施”的办法共同提高系统的可靠性。

（4）易于扩展和升级

考虑到系统的发展和设备的改进，在选择 PLC 容量及 I/O 点数时，应适当留有 20% 左右的余量。

（5）人机界面友好

对于具有人机界面的 PLC 控制系统，应充分体现以人为本的理念。设计的人机操作界面要使用户感到方便、易懂。

### 11.1.2　设计内容

PLC 控制系统的设计内容主要包括硬件选型、设计和软件的编程两个方面，基本由以下几部分组成：

（1）拟定控制系统设计的技术条件

技术条件一般以设计任务书的形式来确定，它是整个控制系统设计的依据。

（2）选择外部设备

根据系统设计要求选择外部输入设备和输出设备。

（3）选定 PLC 的型号

PLC 是整个控制系统的核心部件，合理选择 PLC 对保证系统的技术指标和质量是至关重要的。

（4）分配 I/O 点

根据系统要求，编制 PLC 的 I/O 地址分配表，并绘制 I/O 端子接线图。

（5）设计电气屏柜

根据系统要求，设计操作台、电气柜及非标准电器元件。

（6）软件编写

控制系统的软件包括 PLC 控制软件和上位机控制软件。在编制 PLC 控制软件前要深入了解控制要求与主要控制的基本方法以及系统应完成的动作、自动工作循环的组成、必要的保护和联锁等方面的情况。对比较复杂的控制系统，可利用状态图和顺序功能图方法全面地分析，必要时还可将控制任务分解成几个独立的部分，利用结构化或模块化方法进行编程，这样可化繁为简，有利于编程和调试。

对于有人机界面的 PLC 控制系统，上位机软件的编制也尤为重要。因为上位机软件是系统的操作人员与控制系统之间交互的纽带。良好的人机界面可以让操作人员的操作更为容易，利用上位机软件还能制作历史趋势图、打印报表、记录数据库和故障警报等，使工作效率更加提高。因此，上位机软件的编制十分重要。

（7）系统技术文件的编写

系统技术文件包括说明书、电气原理图、元件明细表、元件布置图、机柜接线图、系统维护手册、上位机软件操作手册、系统安装调试报告等。

### 11.1.3 设计步骤

（1）深入了解和分析被控对象的工艺条件和控制要求

被控对象就是受控的机械、电气设备、生产线或生产过程。控制要求主要是指控制的基本方式、控制指标、应完成的动作、自动工作循环的组成、必要的保护和联锁等。

（2）确定 I/O 设备

根据被控对象对 PLC 控制系统的功能要求，确定系统所需的输入、输出设备。常用的输入设备有按钮、行程开关、选择开关、传感器等，常用的输出设备有继电器、接触器、指示灯、电磁阀、气缸等。

（3）选择合适的 PLC 类型

根据已经确定的 I/O 设备，统计所需要的 I/O 信号的点数，选择 PLC 类型，包括 PLC 机型的选择、容量的选择、I/O 模块的选择、电源模块的选择以及通信模块的选择等。

（4）分配 I/O 点地址

根据所选择的 I/O 模块和其组态的位置分配 PLC 的 I/O 点地址，编制 I/O 点地址分配表并设计输入/输出端子接线图。同时可进行控制柜和操纵台的设计以及现场施工。

（5）设计 PLC 程序

按照系统的控制要求和控制流程要求进行 PLC 程序的设计，其中包括故障的报警和处理方式等。这是整个应用系统设计的核心工作。

（6）PLC 程序的下载与调试

程序设计好后需要通过编程电缆将程序下载到 PLC 的 CPU 中，然后进行软件测试工作。由于在程序的编写过程中难免会有疏漏之处，因此在将 PLC 连接到现场设备之前，一定要先进行软件测试。如果 PLC 程序比较大，最好编写测试程序对程序进行各功能的分段测试。

（7）上位机软件的编程与调试

对于 PLC 控制系统，上位机监控软件的编程与调试也是整个应用系统设计的重点。编程人员根据 PLC 的 I/O 点地址分配表定义上位机软件的地址分配表，并按照系统的控制进程要求设计上位机软件、绘制操作界面。

（8）整个应用系统联调

当现场施工完成，控制柜接线结束，PLC 程序调试通过，且上位机软件编程结束后，就可进行整个应用系统的联合调试。调试过程中应先将主回路脱开，进行控制回路的调试。待控制回路调试一切正常后，再进行带主回路的调试。如果控制系统是由若干个部分组成的，则应先做各局部的调试，然后做整体调试。在系统联调时，不仅仅要做正常控制过程的调试，还应做故障情况的测试，应当尽量将可能的故障情况全部加以测试，确保控制系统的可靠性。

（9）编制技术文件

技术文件是用户将来使用、操作和维护的依据，也是这个控制系统档案保存的重要材料，因此应当给予重视。

以上是一个 PLC 控制系统设计的一般步骤。在具体的应用时，可以根据控制系统的规模、控制流程的繁简程度等情况适当作增减。PLC 控制系统设计流程框图如图 11-1 所示。

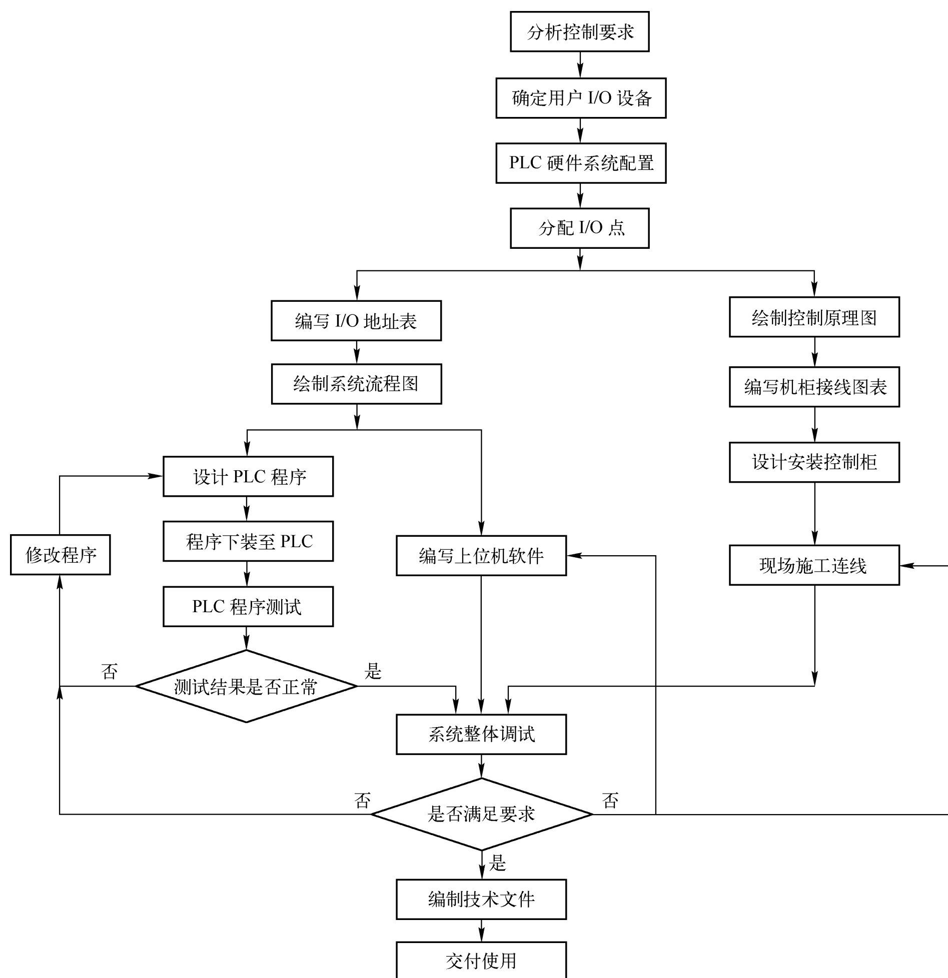



图 11-1　PLC 控制系统设计流程框图

### 11.1.4 硬件设计

1. PLC 的选型

在满足控制要求的前提下，选型时应选择最佳的性能价格比，具体应考虑以下几点：

（1）性能与任务相适应

对于开关量控制的应用系统，当对控制速度要求不高时，可选用小型 PLC（如西门子 S7－200 系列的 PLC 或 OMRON C 系列 P 型机、CPM 型 PLC）就能满足要求。如对小型泵的顺序控制、单台机械的自动控制等。

对于以开关量控制为主，带有部分模拟量控制的应用系统，如工业生产中常遇到的温度、压力、流量、液位等连续量的控制，应选用带有 A/D 转换的模拟量输入模块和带 D/A 转换的模拟量输出模块，配接相应的传感器、变送器（对温度控制系统可选用温度传感器直接输入的温度模块）和驱动装置，并且选择运算功能较强的中型 PLC，如西门子公司 S7－300 系列的 PLC 或 OMRON 公司的 CQM1 型 PLC。

对于控制比较复杂的大中型控制系统，如闭环控制、PID 调节、通信联网等，可选用大、中型 PLC（如西门子公司 S7－400 系列的 PLC 或 OMRON 公司 C200HE/C200HG/C200HX、CV/CCM1 等 PLC）。当系统的各个控制对象分布在不同的地域时，应根据各部分的具体要求来选择 PLC 及远程 I/O 模块，组成一个分散控制系统或远程控制系统。

（2）PLC 的处理速度应满足实时控制的要求

PLC 工作时，从输入信号到输出控制存在着滞后现象，即输入量的变化，一般要在 1～2 个扫描周期之后才能反映到输出端，这对于一般的工业控制是允许的。但有些设备的实时性要求很高，不允许有较大的滞后时间。例如 PLC 的 I/O 点数在几十到几千点范围内，这时用户应用程序的长短对系统的影响速度会有较大的差别。滞后时间应控制在几十毫秒之内，应小于普通继电器的动作时间（普通继电器的动作时间约为 100ms），否则就没有意义了。通常为了提高 PLC 的处理速度，可以用以下几种方法：

① 选择 CPU 处理速度快的 PLC，使执行一条基本指令的时间不超过 0.5μs。

② 优化应用软件，缩短扫描周期。

③ 采用高速响应模块，例如高速计数模块，其影响的时间可以不受 PLC 扫描周期的影响，而只取决于硬件的延时。

（3）PLC 应用系统结构合理、机型系列应统一

PLC 的结构可分为整体式和模块式两种。整体式结构是把 PLC 的 I/O 和 CPU 放在同一块印制电路板上，省去插接环节，体积小，每一 I/O 点的平均价格比模块式的便宜，适用于工艺过程比较稳定、控制要求比较简单的系统；模块式 PLC 的功能扩展，I/O 点数的增减，输入与输出点数的比例，都比整体式方便灵活。维修更换模块、判断与处理故障快速方便，适用于工艺过程变化较多，控制要求复杂的系统。

在使用时，应根据具体情况进行选择。在一个单位或一个企业里，应尽量使用同一系列的 PLC，这不仅使模块通用性好，减少备件量，而且给编程和维修带来极大的方便，也给系统的扩展和升级带来方便。

（4）在线编程和离线编程的选择

小型 PLC 一般使用简易编程器。它必须插在 PLC 上才能进行编程操作，其特点是编程

器与 PLC 共用一个 CPU，在编程器上有一个“运行/监控/编程（RUN/MONITOR/PROGRAM）”选择开关。当需要编程或修改程序时，将选择开关转到“编程（PROGRAM）”位置，这时 PLC 的 CPU 不执行用户程序，只为编程器服务，这就是“离线编程”。当程序编好后再把选择开关转到“运行（RUN）”位置，CPU 则去执行用户程序，对系统实施控制。简易编程器结构简单、体积小，携带方便，很适合在生产现场调试、修改程序用。

图形编程器或者个人计算机与编程软件包配合可实现在线编程。PLC 和图形编程器各有自己的 CPU，编程器的 CPU 可随时对键盘输入的各种编程指令进行处理；PLC 的 CPU 主要完成对现场的控制，并在一个扫描周期的末尾与编程器通信，编程器将编好或修改好的程序发送给 PLC，在下一个扫描周期，PLC 将按照修改后的程序或参数控制，实现“在线编程”。图形编程器价格较贵，但它功能强，适应范围广，不仅可以用指令语句编程，还可以直接用图形编程。目前多使用个人计算机进行在线编程，这样可省去图形编程器，但需要编程软件包的支持，其功能类似于图形编程器。

**2. PLC 容量估算**

PLC 容量包括两个方面：I/O 点数和用户存储器的容量。

（1）I/O 点数的估算

根据被控对象的输入信号和输出信号的总点数，并考虑到今后的调整和扩充，一般应加上 10% ~15% 的备用量。

（2）用户存储器容量的估算

用户应用程序占用多少内存与许多因素有关，如 I/O 点数、控制要求、运算处理量、程序结构等。因此在程序设计之前只能粗略的估算。根据经验，每个 I/O 点及有关功能器件占用的内存如下：

开关量输入：所需存储器字数 = 输入点数 ×10；

开关量输出：所需存储器字数 = 输出点数 ×8；

定时器/计数器：所需存储器字数 = 定时器/计数器数 ×2；

模拟量：所需存储器字数 = 模拟量通道数 ×100；

通信接口：所需存储器字数 = 接口个数 ×300。

根据存储器的总字数再加上一定的备用量。

作为一般应用的经验公式是

所需存储器容量(KB) = (1 ~1.25) × (DI ×10 + DO ×8 + AI/AO ×100 + CP ×300)/1024

其中，DI 为开关量输入总点数；DO 为开关量输出总点数；AI/AO 为模拟量 I/O 通道总数；CP 为通信接口总数。

**3. I/O 模块的选择**

（1）开关量输入模块的选择

PLC 的输入模块用来检测来自现场（如按钮、行程开关、温控开关、压力开关等）的高电平信号，并将其转换为 PLC 内部的低电平信号。

按输入点数分：常用的有 8 点、12 点、16 点、32 点等。

按工作电压分：常用的有直流 5 V、12 V、24 V；交流 110 V、220 V 等。

选择输入模块主要考虑以下两点：

① 根据现场输入信号（如按钮、行程开关）与 PLC 输入模块距离的远近来选择电压的

高低。一般 24 V 以下属低电平，其传输距离不宜太远，如 12 V 电压模块一般不超过 10 m。距离较远的设备选用较高电压模块比较可靠。

② 密度大的输入模块，如 32 点输入模块，能允许同时接通的点数取决于输入电压和环境温度。一般同时接通的点数不宜超过总输入点数的 60%。

（2）开关量输出模块的选择

输出模块的任务是将 PLC 内部低电平的控制信号转换为外部所需电平的输出信号，驱动外部负载。输出模块有 3 种输出方式：继电器输出、双向晶闸管输出、晶体管输出。

① 输出方式的选择。继电器输出价格便宜，使用电压范围广，导通压降小，承受瞬时过电压和过电流能力较强，且有隔离作用。但继电器有触点，寿命较短，且响应速度较慢，适用于动作不频繁的交直流负载。当驱动电感性负载时，最大开关频率不得超过 1 Hz。双向晶闸管输出（交流）和晶体管输出（直流）都属于无触点开关输出，适用于通断频繁的感性负载。感性负载在断开瞬间会产生较高的反压，必须采取抑制措施，如并接阻容吸收电路等。

② 输出电流的选择。模块的输出电流必须大于负载电流的额定值，如果负载电流较大，输出模块不能直接驱动时，应增加中间放大环节。对于电容性负载、热敏电阻负载，考虑到接通时有冲击电流，要留有足够的裕量。

③ 同时接通的输出点数。在选用输出模块时，不但要看一个输出点的驱动能力，还要看整个输出模块的满载负荷能力，即输出模块同时接通点数的总电流值不得超过模块规定的最大允许电流。

（3）特殊功能模块的选择

除了开关量信号以外，工业控制中还要对温度、压力、液位、流量等过程变量进行检测和控制。模拟量输入、模拟量输出和温度控制模块就是用于将过程变量转换为 PLC 可以接收的数字信号，并将 PLC 内的数字信号转换成模拟信号输出。此外，还有一些特殊情况，如步进控制、变速计数以及通信、联网等。与其他外围设备的连接常需要专用的接口模块，如传感器模块、凸轮模块和 PID 模块等。这些模块中有自己的 CPU、存储器，能在 PLC 的管理和协调下独立地处理特殊任务，这样既完善了 PLC 的功能，又可减轻 PLC 的负担，提高处理速度。

**4. 通道分配**

一般输入点与输入信号、输出点与输出设备是一一对应的。程序设计前，应按系统配置的通道与触点号，分配给每一个输入信号和输出信号，即进行通道分配。

在个别情况下，也有两个信号用一个输入点的，那样就应在接入输入点前，按逻辑关系接好线（如两个触点先串联或并联），然后再接到输入点。

（1）明确 I/O 通道范围

不同型号的 PLC，其输入/输出通道的范围是不一样的，应根据所选 PLC 型号，查阅相应的编程手册，决不可“张冠李戴”。

（2）内部辅助继电器

内部辅助继电器不对外输出，不能直接连接外部器件，而是在控制其他继电器、定时器/计数器时作数据存储或数据处理用。从功能上讲，内部辅助继电器和数据存储器相当于传统电控柜中的中间继电器。根据程序设计的要求，应合理安排 PLC 的内部辅助继电器和数据

存储器，在设计说明书中应详细列出各内部辅助继电器和数据存储器在程序中的用途，避免重复使用。

（3）分配定时器/计数器

程序中使用到的定时器/计数器的编号不能相同。若扫描时间较长，应使用高速定时器，以确保计时准确。

（4）数据存储器

在数据存储、数据转换以及数据运算等场合，经常需要以通道为单位的数据，此时，应用数据存储器是很方便的。数据存储器中的内容，即使在 PLC 断电、运行开始或停止时也能保持不变。数据存储器也应根据程序设计的需要来合理安排，需详细列出各数据存储器通道在程序中的用途，以避免重复使用。

**5. 外部接线设计**

在 PLC 选型和通道分配结束后，可根据手册的规定画出 PLC 外部接线，主要包括以下内容：

（1）电源

PLC 通常采用 220 V 交流电源，允许一定的波动，但为了提高系统的可靠性，应在输入端配置 1∶1 隔离变压器（如果电网电压过高，为了安全，可取 1∶0.9），且应有独立使用的断路器。

（2）接地

PLC 在大多数情况下可以不做接地处理，但在条件允许时应尽量设计接地线路。在实际 PLC 控制系统中，接地是抑制干扰、使系统可靠工作的主要方法。在设计中如能把接地和屏蔽正确地结合起来使用，可以解决大部分干扰问题。

一般情况，PLC 控制系统应设置独立接地，为保证接地质量，一般要求接地电阻应小于 4 Ω，实在做不到，也可与弱电系统共地。在噪声较大时，可将噪声滤波端与接地端短接。

（3）输入

PLC 可与有触点及电流型输入设备相连，但不能与电压型输入设备相连接。在 I/O 接线设计时，应检查所有输入设备的兼容性，并充分考虑漏电流和负载感应电动势的影响。

（4）输出

晶体管或双向晶闸管输出型 PLC 接上负载后，当漏电流较大有可能造成设备的误动作时，应在负载两端并联一个旁路电阻，如图 11–2 所示。

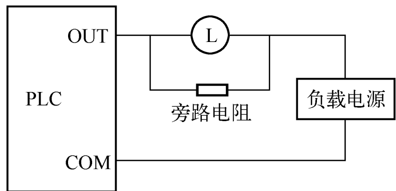


图 11–2　负载并联旁路电阻

当感性负载连到 PLC 输出端时，同样需要加电涌抑制器或二极管，用以吸收负载产生的反电动势，如图 11–3 所示。其中，二极管必须耐 3 倍的负载电压，并允许流过 1 A 的平均电流。当负载电压为 220 V 时，阻容吸收装置中 R = 50 Ω，C = 0.47 μF。

还需要注意的是，对于易造成伤害事故的负载。除了在 PLC 的控制程序中要加以考虑外，还应在 PLC 之外设计急停电路，设置事故开关、紧急停机装置等，使得一旦设备发生故障时，能及时切断引起伤害事故的负载电源。

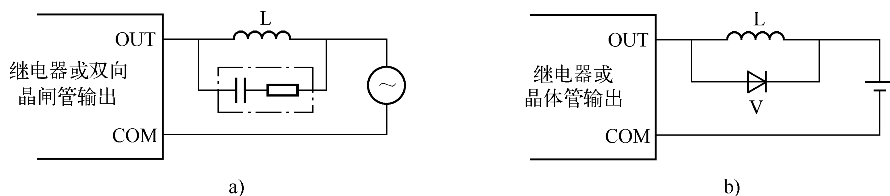



图 11-3　感性负载输出

a）继电器或双向晶闸管输出　b）继电器或晶体管输出

### 11.1.5　软件设计

根据 PLC 系统硬件结构和生产工艺要求，使用相应的编程语言，编制实际应用程序，并形成程序说明书的过程就是软件设计。

**1．程序设计前的准备工作**

（1）了解系统概况，形成整体概念

这一步工作主要是通过系统设计方案了解控制系统的全部功能、控制规模、控制方式、输入和输出信号的种类及数量、是否有特殊功能接口、与其他设备的关系、通信内容与方式等。如果没有对整个控制系统的全面了解，就不能对各种控制设备之间的相互联系有真正的理解，靠想当然编制的程序是肯定无法实际运行的。

（2）熟悉被控对象，使程序设计有的放矢

将控制要求根据控制功能分类，并确定输入信号检测设备、控制设备、输出信号控制装置的具体情况，深入细致地了解每一个检测信号和控制信号的形式、功能、规模，它们之间的关系，并预见以后可能出现的问题，使程序设计有章可循。

在熟悉被控对象的同时，还要认真借鉴前人在程序设计中的经验和教训，总结各种问题的解决方法。总之，在程序设计之前，掌握的东西越多，对问题思考得越深入，程序设计就会越得心应手。

（3）充分地利用各种软件编程环境

目前各 PLC 主流产品都配置了功能强大的编程环境，如西门子公司的 STEP 7、欧姆龙公司的 CX-P 软件等，可在很大程度上减轻软件编制的工作强度，提高编程效率和质量。

**2．程序框图设计**

程序框图设计工作主要是根据控制系统的具体情况，确定用户程序的基本结构、程序设计标准和结构框图，然后再根据工艺要求，绘制出各个功能单元的详细功能框图。系统程序框图应尽量做到模块化，一般最好按功能采取模块化设计方法，因此相应的框图也应依此绘制，并规定其各自应完成的功能，然后再绘制图中各模块内部的详细功能、顺序功能图和状态图。框图是编程的主要依据，要尽可能准确和详细。如果框图是由别人设计的，一定要设法弄清楚其设计思想和方法。完成这部分工作之后就会对系统全部程序设计内容具有一个整体思想，为下一步的程序设计奠定良好的基础。

**3．编写程序**

编写程序就是根据设计出的顺序功能或状态图编写控制程序，这是整个程序设计工作的核心部分。

在编写程序的过程中，可以借鉴典型的标准程序，但必须读懂这些程序段，否则将会给后续工作带来困难和损失。另外，编写程序过程中要编写符号表，并及时对编写出的程序进

行注释，以免忘记相互之间关系。

4. 程序测试

刚编好的程序难免存在错误或缺陷。为了及时发现和消除程序中的错误和缺陷、减少系统现场调试的工作量、确保系统在各种正常和异常情况时都能做出正确的响应，需要对程序进行离线测试。经调试、排错、修改及模拟运行后，才能正式投入运行。

程序测试时重点应注意下列问题：

① 程序能否按设计要求运行。

② 各种必要的功能是否具备。

③ 发生意外事故时能否做出正确的响应。

④ 对现场干扰等环境因素适应能力如何。

经过测试、排错和修改后，程序基本正确，下一步就可到使用现场试运行，进一步察看系统整体效果，还有哪些地方需要进一步完善。经过一段时间运行，证明系统性能稳定，工作可靠，已达到设计要求，就可把程序下装到 PLC 的 CPU 中，正式投入运行。

5. 编写程序说明书

程序说明书是程序设计的综合说明文档。编写程序说明书的目的是为了便于程序的设计者与现场工程技术人员进行程序调试与程序修改工作，它是程序文件的组成部分。程序说明书一般应包括程序设计的依据、程序设计与调试的关键点等。

### 11.1.6 PLC 控制系统的抗干扰设计

PLC 属于专用工业控制计算机，一般放置和工作于工业现场，直接与被控装置及设备相连接，由于其自身工作电压较低，工作频率较高，因此现场的各种干扰对它将会造成很大的影响，甚至引起误动作造成重大的损失。特别是系统的输入、输出环节，更是干扰进入的重要通道，因此在 PLC 控制系统设计中，考虑相应的抗干扰措施是极为重要的。工业现场环境条件一般比较恶劣，为此必须考虑 PLC 控制系统的合理抗干扰措施。

1. 抑制公共阻抗耦合干扰的措施

导线的阻抗通常就是耦合阻抗，其大小与导线的敷设有很大关系，在电路或系统设计时必须预先考虑抗干扰措施。主要如下：

① 尽量缩短公共阻抗部分的导线长度、减小来回线间的距离及采用直线布线方式等，用以减小导线的电感。

② 采取增大导线截面、减小接触电阻等措施减小导线的电阻。

③ 机柜接地与系统接地分开设置。

④ 采用继电器、光耦合器、变压器等电隔离器件实现电位隔离。

2. 抑制电容性干扰的措施

为了抑制和避免电容性干扰，在设计 PLC 控制系统时，要求干扰源的电气参数应使电压变化幅度和变化速率尽可能的小；被干扰系统应尽可能设计成低电阻及高信噪比系统，并且其结构应尽量紧凑，且彼此在空间上相互隔离。

① 为减小耦合电容，在配线时，应使导线尽量短些，并尽量避免平行走线，信号线必须与电源线分别敷设；弱电线与强电线分别安排在不同配线槽内等。

② 通过电气屏蔽抑制干扰源的干扰电场的作用。

③ 将耦合电容彼此电气对称、平衡地连接，可以抵消耦合的干扰作用。

3. 抑制电感性干扰的措施。

抑制电感性干扰的措施主要有三种方法：

① 减小系统各部分之间的电感。主要措施是减小系统各单元耦合部分（主要是电线电缆）的间距、导线尽量短、避免平行走线、采用双绞线以缩小电流回路所围成的面积等。

② 对干扰对象或干扰源设置磁屏蔽，以抑制干扰电场。主要有静态磁屏蔽和涡流屏蔽等措施。静态磁屏蔽主要用于低频段，屏蔽对象用一个尽可能密闭的铁磁性外壳罩起；涡流屏蔽用在高频段，它是利用非磁性或弱磁性物质中的涡流效应对交变磁场进行屏蔽。

③ 采用结构平衡措施。如导线垂直交叉敷设，或采用双绞线结构等，使耦合的干扰信号最小，或彼此抵消。

4. 抑制波阻抗耦合干扰的措施

波阻抗耦合分为传导波耦合和辐射波耦合两种形式。对于传导波耦合，可通过强电线与弱电线分开敷设，使用双绞线或同轴屏蔽电缆等措施加以抑制；对于辐射波干扰，通常采用在干扰源和干扰对象间插入金属屏蔽物的方式来抑制。

## 11.2 系统调试与检查

PLC 为系统调试提供了强大的功能，充分利用这些功能，将使系统调试简单、迅速。

### 11.2.1 系统调试步骤

系统调试时，应首先按要求将电源、I/O 端子等外部接线连接好，然后将已编好的程序下载到 PLC，并使其处于监控或运行状态。系统调试流程如图 11-4 所示。

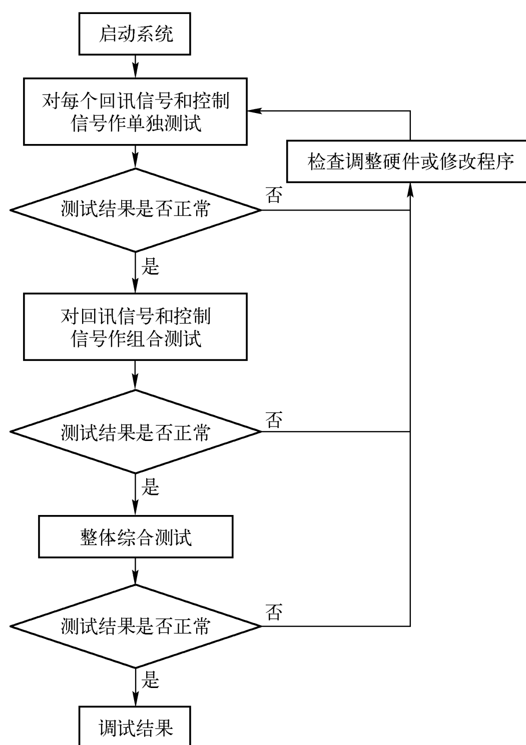



图 11-4 系统调试流程图

### 11.2.2 系统调试方法

**1. 对每个现场信号和控制量作单独测试**

对于一个系统来说，现场信号和控制量一般不止一个，但可以人为地使各现场信号和控制量一个一个单独满足要求。当一个现场信号和控制量满足要求时，观察 PLC 输出端和相应的外部设备的运行情况是否符合系统要求。如果出现不符合系统要求的情况，可以先检查外部接线是否正确，当接线准确时再检查程序，修改控制程序中的不当之处。直到每一个现场信号和控制量单独作用时均满足系统要求为止。

**2. 对现场信号和控制量作模拟组合测试**

通过现场信号和控制量的不同组合来调试系统，也就是人为地使两个或多个现场信号和控制量同时满足要求，然后观察 PLC 输出端及外部设备的运行情况是否满足系统控制的要求。一旦出现问题（基本上属于程序问题)，应仔细检查程序并加以修改，直到满足系统要求为止。

**3. 整个系统综合调试**

整个系统的综合调试是对现场信号和控制量按实际控制要求进行模拟运行，以观察整个系统的运行状态和性能是否符合系统的控制要求。若控制规律不符合要求，绝大多数是因为控制程序有问题，应仔细检查并修改控制程序；若性能指标不满足要求，应该从硬件和软件两方面加以分析，找出解决办法。调整硬件或软件，使系统达到控制要求。

## 11.3 交流电动机正、反转控制的工程应用方法

现在用户已经具备一定的编程基础，本节内容可以让用户从工程的角度对控制有所了解。

在第 4 章，用 PLC 控制交流电动机的正、反转方法无疑是正确的。但是，在实际工程应用中，需要考虑的因素就没那么简单了。现在越来越多的大中型项目会利用计算机作为远程上位机来监控现场设备的运行，而且实际工程情况复杂多变，所以要尽可能全面细致地考虑问题，以实现控制操作的安全性和准确性。

以下的情况在实际工程中是常见的，而在图 4-12 的程序中不能处理这些情况：

1）在实际复杂的情况下，比如 PLC 故障，电动机一旦转动便无法用停止按钮使电动机停下来，这时只能去切断主回路电源才能使电动机停止运转。这时，手动控制必须要有最高的优先权，也就是说，用户希望在任何情况下按下“停止按钮”，电动机都会停止运转。另外，如果 PLC 不在“运行”状态，则当按下电动机运转按钮后，PLC 的输出端就不会有输出信号，使电动机无法正常运转。

2）按下“正转按钮”希望电动机开始正转，但由于控制正转的交流接触器出现故障，电动机并没有动作，如果此时控制按钮距离现场设备较远，由于没有电动机运行状态的反馈信号，PLC 就不能准确地判断此时的电动机运行状态。

由此我们知道，在上面例子中实现电动机正、反转的 PLC 程序是不具备工程实用意义的。在工程设计中还必须考虑系统的故障及执行器的反馈回讯信号。

## 11.3.1　工程应用基础

在完整的实际工程项目中，设计工程师主要致力于两部分的工作：一是 PLC 硬件系统的设计和编程；二是利用组态软件实现远程的计算机监控。两部分工作须紧密结合、相辅相成。

组态软件是指一些数据采集与过程控制的专用软件，它们是在自动控制系统监控层一级的软件平台和开发环境。组态软件具有灵活的组态方式，是为用户提供快速构建工业自动控制系统监控功能的通用层次上的软件工具。组态软件能支持各种工控设备和常见的通信协议，并且通常应提供分布式数据管理和网络功能。对应于原有的 HMI（人机接口软件，Human Machine Interface）的概念，组态软件应该是一个使用户能快速建立自己的人机界面（HMI）的开发环境。当然软件中的组态要比硬件的组装有更大的发挥空间，因为它一般要比硬件中的“部件”更多，而且每个“部件”都更灵活，因为软部件都有其内部属性，通过改变属性可以改变其规格（如大小、性状、颜色等）。所以，利用组态软件制作的上位机监控很形象，用户可以直接通过鼠标在计算机屏幕上操作，就能完成控制任务，同时不必亲临操作现场也可以掌握现场详细情况。

现在以这种工程应用思路，介绍 PLC 控制系统的工程应用编程思想，即上位机和下位机的联合工作。

首先我们要理解两个工程上的常用术语：回讯信号和控制信号。

回讯信号是现场执行器件是否有动作的反馈信号，反映其运行状态。比如第 4 章举例的电动机正、反转线圈 KM1 和 KM2，利用这两个线圈的常开触点间接表明执行器电动机是否处于“运行”状态。回讯信号一般由现场设备（如继电器、交流接触器等）的辅助触点接入到 PLC 数字量输入模块（通常使用 DC 24V 电源驱动）。本例也需如此，只有捕捉到电动机运行的回讯信号，才能表明电动机运行基本正常。否则，必须在上位机进行故障指示，或执行相应的处理动作。

控制信号是用于使现场设备动作的信号。控制信号一般通过上位机由 PLC 的数字量模块的输出点接入到现场设备的执行机构或中间继电器线圈。

上位机和下位机的配合、控制和回讯信号的体现如图 11-5 所示。

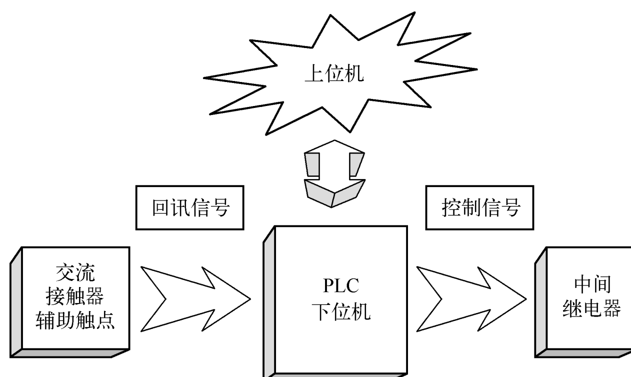



图 11-5　上位机和下位机的配合、控制和回讯信号的体现

## 11.3.2　控制原理

系统的电气主回路原理图如图 11-6 所示。

电动机正、反转手动/自动控制回路原理如图 11-7a、b 所示。

图 11-7a 和图 11-7b 中，KM1 和 KM2 分别代表电动机正转和反转的交流接触器；JDQ1、JDQ2、JDQ3 分别代表正转、反转和停止的中间继电器。由于选用了中间继电器（线圈 DC 24V，触点 AC 220V），所以 PLC 系统的输入和输出模块都选用 DC 24V 模块。图 11-7a 中，PLC 的输入端分别接入正转、反转交流接触器和主回路电源断路器的常开辅助触点，作为交流接触器和主回路电源的回讯信号，反映电源、电动机正转和电动机反转的状态；PLC 的输出端，分别接至正转、反转和停止继电器的线圈，控制继电器动作。

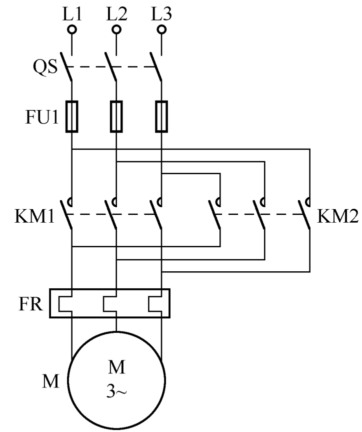



图 11-6　系统的电气主回路原理图

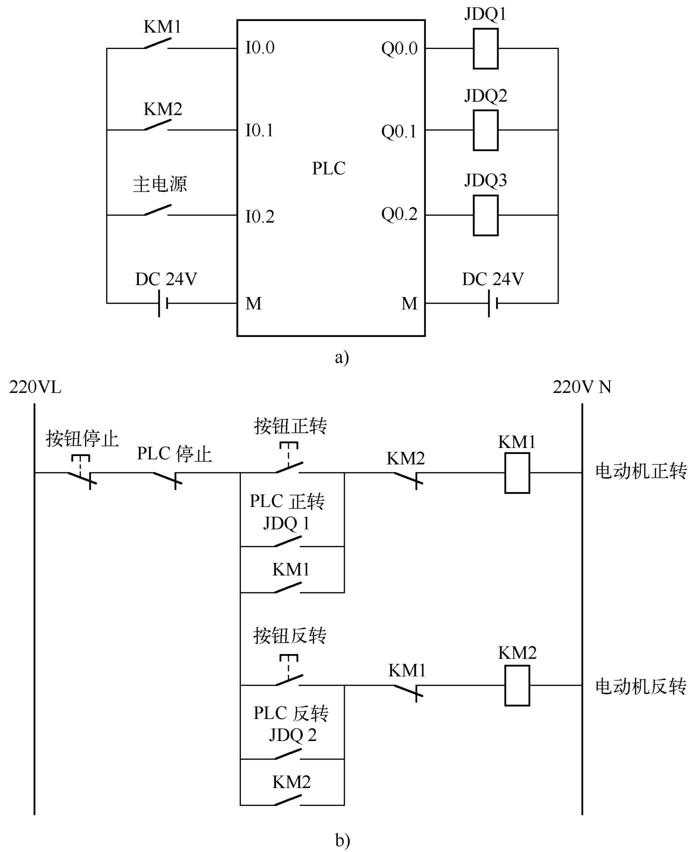



图 11-7　电动机正、反转手动/自动控制回路原理图

图 11-7b 完成电动机正、反转的手动及自动控制。手动控制是通过纯硬件电路实现的。自动控制是上位机通过 PLC 对中间继电器进行控制，取中间继电器的常开触点作为输出线圈。图中手动“正转按钮”与中间继电器的常开触点“PLC 正转 JDQ1”为并联关系，表示

手动方式和自动方式起同样的作用。当上位机发出正转命令后，PLC 的输出点 Q0.0 产生高电平信号，驱动中间继电器 JDQ1 线圈，图 11-7b 中 JDQ1 的常开触点“PLC 正转 JDQ1”闭合，使控制回路接通，交流接触器 KM1 得电，此时 KM1 的常开辅助触点闭合，使电动机主回路中正转回路保持接通状态。由于回路具有自锁功能，此时即使“PLC 正转 JDQ1”触点释放，电动机仍能保持正转运行状态。电动机反转的自动控制原理与正转过程相同。

电动机正、反转控制程序变量如表 11-1 所示。在前述的 PLC 控制系统，PLC 有三个输入点，分别连接“正转按钮”、“反转按钮”和“停止按钮”；两个输出点，分别连接控制正转和反转的中间继电器。

**表 11-1　电动机正、反转控制程序变量表**

| PLC 地址 | 上位机地址 | 变　量　名 | 意　　义 |
|---|---|---|---|
| I0.0 | | 电动机正转 | 回讯信号 |
| I0.1 | | 电动机反转 | 回讯信号 |
| I0.2 | | 主回路电源 | 回讯信号 |
| Q0.0 | M10.0 | 电动机正转 | 上位机控制信号 |
| Q0.1 | M10.1 | 电动机反转 | 上位机控制信号 |
| Q0.2 | M10.2 | 电动机停止 | 上位机控制信号 |
| M5.0 | M5.0 | 释放正转命令 | 中间变量 |
| M5.1 | M5.1 | 释放反转命令 | 中间变量 |
| M5.2 | M5.2 | 释放停止命令 | 中间变量 |

而实际工程中，除去电源的回讯信号，还有两个交流接触器辅助触点的回讯信号。所以输入点有三个；输出点则为上位机通过 PLC 的控制信号：有电动机正转、电动机反转和电动机停止三个输出。

表 11-1 中还有释放命令 M5.0 ~ M5.2。释放中间继电器的作用有两个方面：其一是为了延长继电器的触点寿命，最好不要长期通电吸合；其二避免当硬件出现故障时电动机没有动作，而当故障排除后电动机产生的误动作。

上位机和 PLC 联合控制的流程如图 11-8 所示。工程应用方法的 PLC 控制程序如图 11-9 所示。

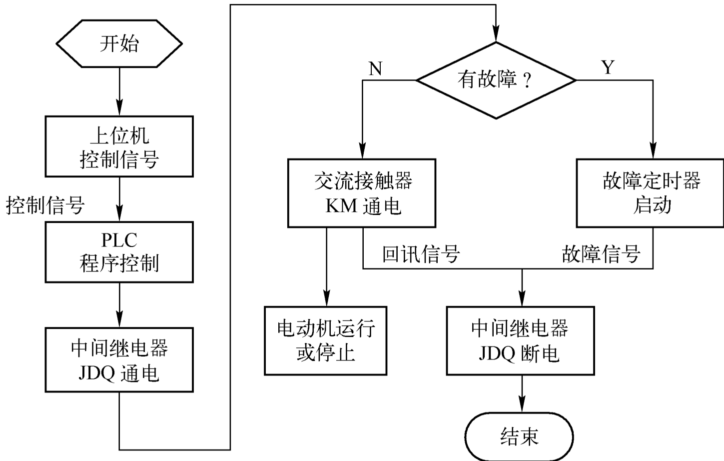
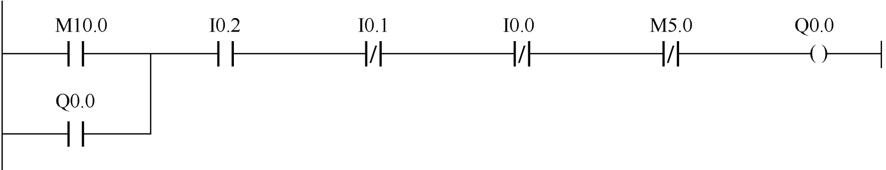


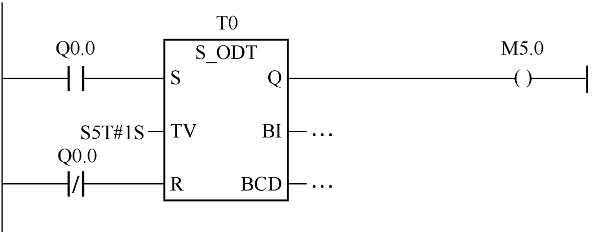

图 11-8　上位机和 PLC 联合控制的流程图

OB1："WINCC控制电动机正、反转程序"

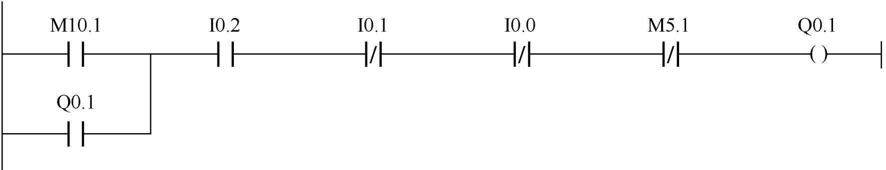
Network 1：电动机正转控制（M10.0：上位机正转命令）



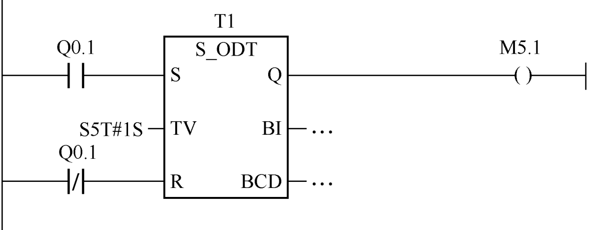

Network 2：延时1s后将继电器（Q0.0）动作释放



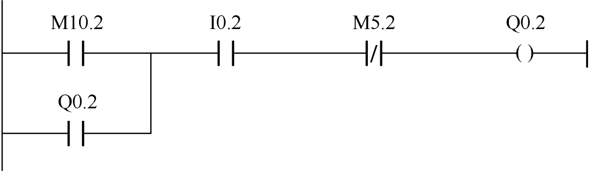

Network 3：电动机反转控制（M10.1：上位机反转命令）




Network 4：延时1s后将继电器（Q0.1）动作释放




Network 5：电动机停止控制（M10.2：上位机停止命令）




Network 6：延时1s后将继电器（Q0.2）动作释放

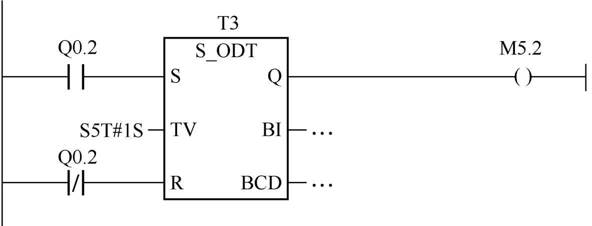



图 11-9　工程应用方法的 PLC 控制程序

Network 7：消上位机命令

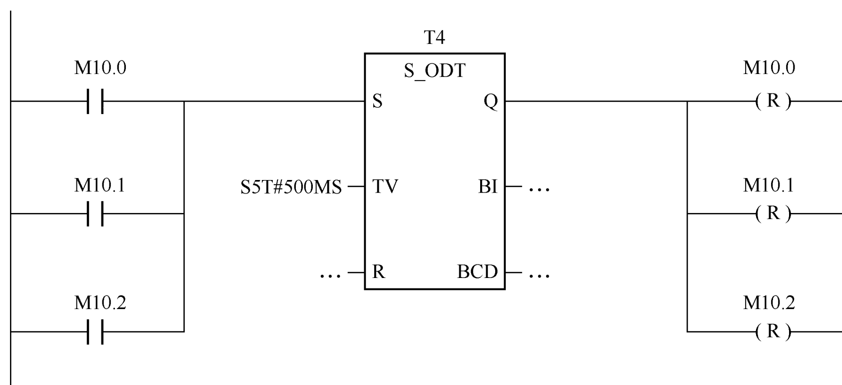



图 11-9　工程应用方法的 PLC 控制程序（续）

说明：程序中 Network1、Network3 和 Network5 是实现自动控制要求的基本程序，在程序中，M10.0、M10.1 和 M10.2 分别是控制正转、反转和停止的点动作，它们的信号来自上位机画面中的“正转”、“反转”和“停止”按钮。当总电源接通（I0.2 闭合），在上位机画面中按下“正转”按钮时，程序中 Network1 的 M10.0 闭合，Q0.0 得电，程序实现自锁；同时使中间继电器 JDQ1 闭合，KM1 得电，它的常开触点闭合，自动控制的主回路正转电路接通（见图 11-7b），电动机正转。与此同时使电动机正转的回讯信号 I0.0 得电（见图 11-7a），由于在程序中 Network1 串入 I0.0 的常闭触点，所以此时 Q0.0 的输出信号随着回讯信号的到来而释放，保证了中间继电器的短时接通。为了加强控制的安全性，又在程序中编写了其余的 4 个网络。

在程序的 Network2、Network4 和 Network6 中，都使用了延时为 1s 的定时器，它们起到看门狗一样的作用。解决了这样一个实际故障，即当上位机发出命令后，PLC 没有正常收到相应动作的回讯信号（可能是由于交流接触器故障、线路故障等硬件原因），而此时的输出命令仍然处于自保持状态而没有断开，容易引发误动作。所以，需要及时将控制信号清除，实现在控制信号得电 1s 后使相应的 M5.0 ~ M5.2 通电，从而使段落 Network1、Network3 和 Network5 自动切断自保持，这样使中间继电器短时接通。

在 Network7 中，当上位机发出指令 0.5s 后，PLC 自动清除上位机命令，以避免控制回路中产生误动作。这是编程的好习惯。

上位机控制画面如图 11-10 所示。图中有“正转”、“反转”和“停止”按钮。同时有虚拟的电动机正、反转控制主回路。在正转时，正转线圈通电，反映在相应的乒乓开关在闭合状态，同时在输入/输出域上显示“正转”。反转同理。在故障发生后，可以在报警记录的故障消息中归档处理。

通过本例可以看出工程应用时考虑系统的正常、可靠运行是至关重要的。

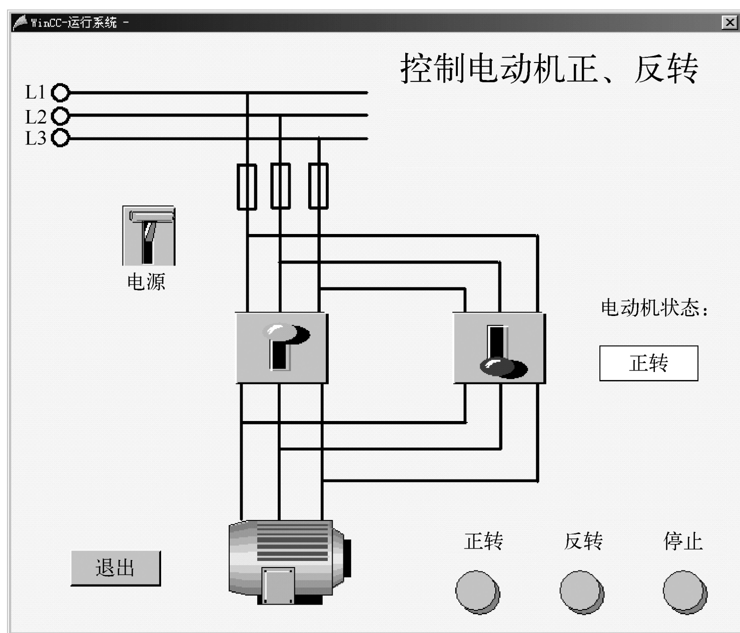


图 11-10　上位机控制画面

## 11.4　闸门自动监控系统工程实例

下面我们将通过 PLC 在工业应用系统中的实例对 PLC 的设计过程作全方位的了解。引用的实例是一个闸门自动化监控系统。

### 11.4.1　项目概况和要求

本实例中的闸门是位于河道上的船闸，分为上下游两个闸首，每个闸首各由一个单孔闸门构成，船闸闸室长 140 m，宽 16 m。为充分发挥工程效益和提高工程运行管理水平，建设闸门自动化监控系统。

该船闸自动化监控系统工程建设的目标是通过广泛应用现代计算机技术、通信技术、自动控制技术、微机保护技术，建立闸站的自动控制系统，为该船闸的现代化管理提供一个坚实的平台。该船闸自动化系统建成后能达到如下要求：

1）可以实时完整地完成对闸站的两台闸门系统及其他辅助设备等信息的收集、处理和存储，建立完整的数据库。主要对上、下闸首闸门（启闭机等）进行控制，并控制通航指挥系统的上、下闸首的通航信号灯等设备；测量上游、下游及闸室水位值，采集配电柜内的电压、电流值。

2）可以在中心控制室的上位机上远程启闭闸门及控制其他相关设备，在现地监控机柜上设有手动/自动，现地/远方等控制方式，基本达到无人值班的要求。

3）具有良好的人机界面，能向管理员提供系统可靠的运行工况和参数，方便管理员的控制操作，上位机具有历史数据存储、查询和统计功能。

### 11.4.2 系统总体设计

闸门自动化监控系统主要是运用可编程序控制器、计算机网络、光纤通信等先进技术对闸站内设备进行自动控制。本系统能实现实时信息自动采集、传输、处理入库、动态监测监控、远程数据传输等功能，实现闸门控制自动化，大大地提高水利设施的运行管理水平。

本系统主要是对闸站内设备进行自动控制，根据工程范围及系统要求达到的目标，本闸门自动化监控系统主要负责监控以下设备：监控对象包括两套闸门启闭机设备、闸位计、闸门荷重、4 套通航灯设备、上下游及闸室水位计、采集配电柜内的电压、电流值等。

整个监控系统通过以太网构成，系统主要由两套 PLC 现地监控柜、水位传感器、闸位传感器、电流检测单元、闸门荷重仪、监控计算机、激光打印机、网络交换机、不间断电源等设备组成。

监控中心在本工程中配置了 1 套以太网交换机，负责系统局域网内所有工作站及现地监控单元的连接；控制室中央布置了工作台，台上放置监控主机，负责闸站内设备的自动监测和控制工作；1 台打印机，打印日常数据报表；工作台下放置计算机的主机；系统配置了 1 套 UPS 电源，负责整个监控中心内设备的供电。

整个系统网络拓扑结构采用层次式星形结构，以监控中心以太网交换机为中心，通过屏蔽超五类网络线和光纤连接到现地单元的 LCU 及工作站。

系统共有 2 套 PLC 现地监控单元，其中上闸首现地控制柜内放置 1 套 PLC，对上闸首启闭机等设备进行实时控制；下闸首现地控制柜内放置 1 套 PLC，对下闸首启闭机等设备进行实时控制。

系统拓扑结构示意图如图 11-11 所示。

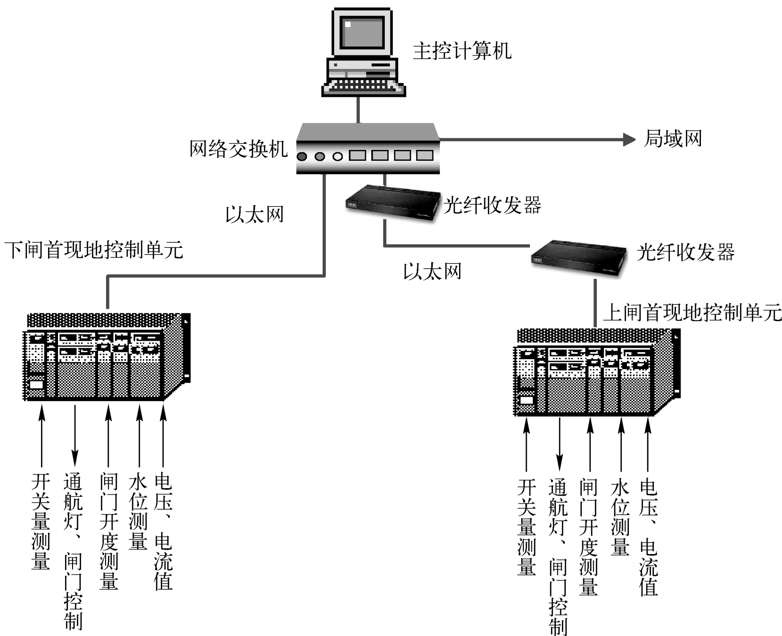



图 11-11　系统拓扑结构示意图

闸门控制的设计主要是：闸门的上升、下降和停止；空气断路器的紧急切断；闸门检测信号有：闸门开高、闸门上限位置、闸门下限位置；闸门控制回讯信号有：闸门上升回讯、闸门下降回讯、闸门控制手/自动回讯、电源开关合回讯。

闸门手动控制箱具有现场手动控制、同时接受现地控制单元的控制并向 PLC 提供回讯信号的接口。

### 11.4.3 PLC 模块及其他设备的选型

#### 1. 系统点数统计

由系统拓扑图可知，本系统在上、下闸首各有 1 套 PLC 控制柜。各套 PLC 根据系统控制设计的要求连接的设备主要有：

（1）数字量输入信号

闸门回讯信号共有 7 点；通航灯回讯信号共有 4 点；闸室外水位编码器采用浮子式水位计，输入 12 位格雷码信号共有 12 点。

（2）数字量输出信号

闸门控制信号共有 4 点；通航灯控制信号共有 4 点。

（3）模拟量输入通道

闸室水位 1 路 4 ~ 20 mA 电流信号；闸门荷重 1 路 4 ~ 20 mA 电流信号；电流采集表 2 路 4 ~ 20 mA 电流信号。

（4）其他

闸位计采用德国海德汉公司的绝对值旋转编码器，编码器通过同步串行接口输出格雷码信号的定位数据值，因此需要 1 路 SSI 信号输入模块。

系统中每套 PLC 单元所需具体使用的点数统计如表 11-2 所示。

表 11-2　各套 PLC 单元具体使用的点数统计表

| 序　　号 | 信号类型 | 点　　数 |
|---|---|---|
| 1 | 数字量输入点 | 23 点 |
| 2 | 数字量输出点 | 8 点 |
| 3 | 模拟量输入 | 4 路（4 ~ 20 mA） |
| 4 | SSI 信号 | 1 路 |

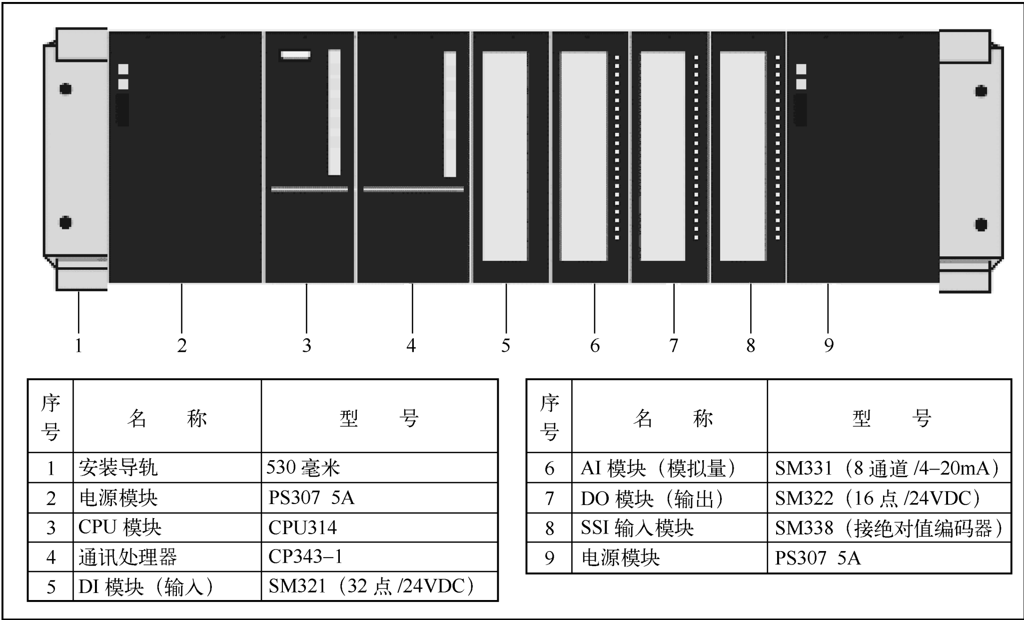
#### 2. PLC 的选型

本系统共需要 2 套 PLC 模块，且 2 套 PLC 模块配置相同。本系统的 PLC 模块均选用西门子 S7 - 300 系列模块。系统的 PLC CPU 模块选用 CPU314 模块（配 MMC 64K Flash 存储卡）；数字量输入模块选用 SM321（32 点/DC 24V）；数字量输出模块选用 SM322（16 点/DC 24V）；模拟量输入模块选用 SM331（8 通道/4 ~ 20 mA）；闸位计编码器输出 SSI 信号，因而选用 SSI 输入模块 SM338。另外，本系统的上位机与 PLC 通过以太网通信，因此选用 CP343 - 1 模块作为以太网通信模块。通过选型，PLC 的最终设备如表 11-3 所示。

表 11-3　各套 PLC 单元具体使用的模块及设备

| 序　号 | 名　　称 | 型　　号 | 定　货　号 | 单位 | 数量 |
|---|---|---|---|---|---|
| 1 | CPU | CPU314 | 6ES7 314 - 1AF10 - 0AB0 | 套 | 2 |
| 2 | Flash 卡 | MMC 64K | 6ES7 953 - 8LF11 - 0AA0 | 块 | 2 |
| 3 | 安装导轨 | 530mm | 6ES7 390 - 1AF30 - 0AA0 | 块 | 2 |
| 4 | 电源模块 | PS307　5A | 6ES7 307 - 1EA00 - 0AA0 | 块 | 4 |
| 5 | A/D 模块（模拟量） | SM331（8 通道/4 ~ 20mA） | 6ES7 331 - 7KF02 - 0AB0 | 套 | 2 |
| 6 | I/O 模块（输入） | SM321（32 点/DC 24V） | 6ES7 321 - 1BL00 - 0AA0 | 套 | 2 |
| 7 | DO 模块（输出） | SM322（16 点/DC 24V） | 6ES7 332 - 1BH01 - 0AA0 | 套 | 2 |
| 8 | SSI 输入模块 | SM338（接绝对值编码器） | 6ES7 338 - 4BC01 - 0AB0 | 套 | 2 |
| 9 | 通信处理器 | CP343 - 1 | 6GK7 343 - 1EX20 - 0AA0 | 块 | 2 |
| 10 | 20 针连接器 | | 6ES7 392 - 1AJ00 - 0AA0 | 套 | 6 |
| 11 | 40 针连接器 | | 6ES7 392 - 1AM00 - 0XAA0 | 套 | 2 |
| 12 | 编程适配器 | | 6ES7 972 - 0CA23 - 0XA0 | 套 | 1 |
| 13 | PLC 编程软件 | STEP - 7 | | 套 | 1 |

根据所选的模块，对 PLC 各模块位置安排如图 11-12 所示。




| 序号 | 名　称 | 型　号 |
|---|---|---|
| 1 | 安装导轨 | 530 毫米 |
| 2 | 电源模块 | PS307　5A |
| 3 | CPU 模块 | CPU314 |
| 4 | 通讯处理器 | CP343-1 |
| 5 | DI 模块（输入） | SM321（32 点 /24VDC） |

| 序号 | 名　称 | 型　号 |
|---|---|---|
| 6 | AI 模块（模拟量） | SM331（8 通道 /4-20mA） |
| 7 | DO 模块（输出） | SM322（16 点 /24VDC） |
| 8 | SSI 输入模块 | SM338（接绝对值编码器） |
| 9 | 电源模块 | PS307　5A |

图 11-12　PLC 各模块位置安排

## 11.4.4　控制原理图及设备接线图的设计

本监控系统中的 2 套 PLC 系统各自形成相对独立的系统，所控制的设备较少，因此系统控制原理图设计也不复杂。只需设计出 1 套系统控制原理图即可。

本监控系统中控制闸门的上升、下降动作，实质上是控制闸门启闭机的正转、反转动作。闸门启闭机为 AC 380V 三相异步电动机。系统主回路及控制回路原理图如图 11-13 所示，上下游通航灯控制原理图如图 11-14 所示，图中各标记的意义参见表 11-4。

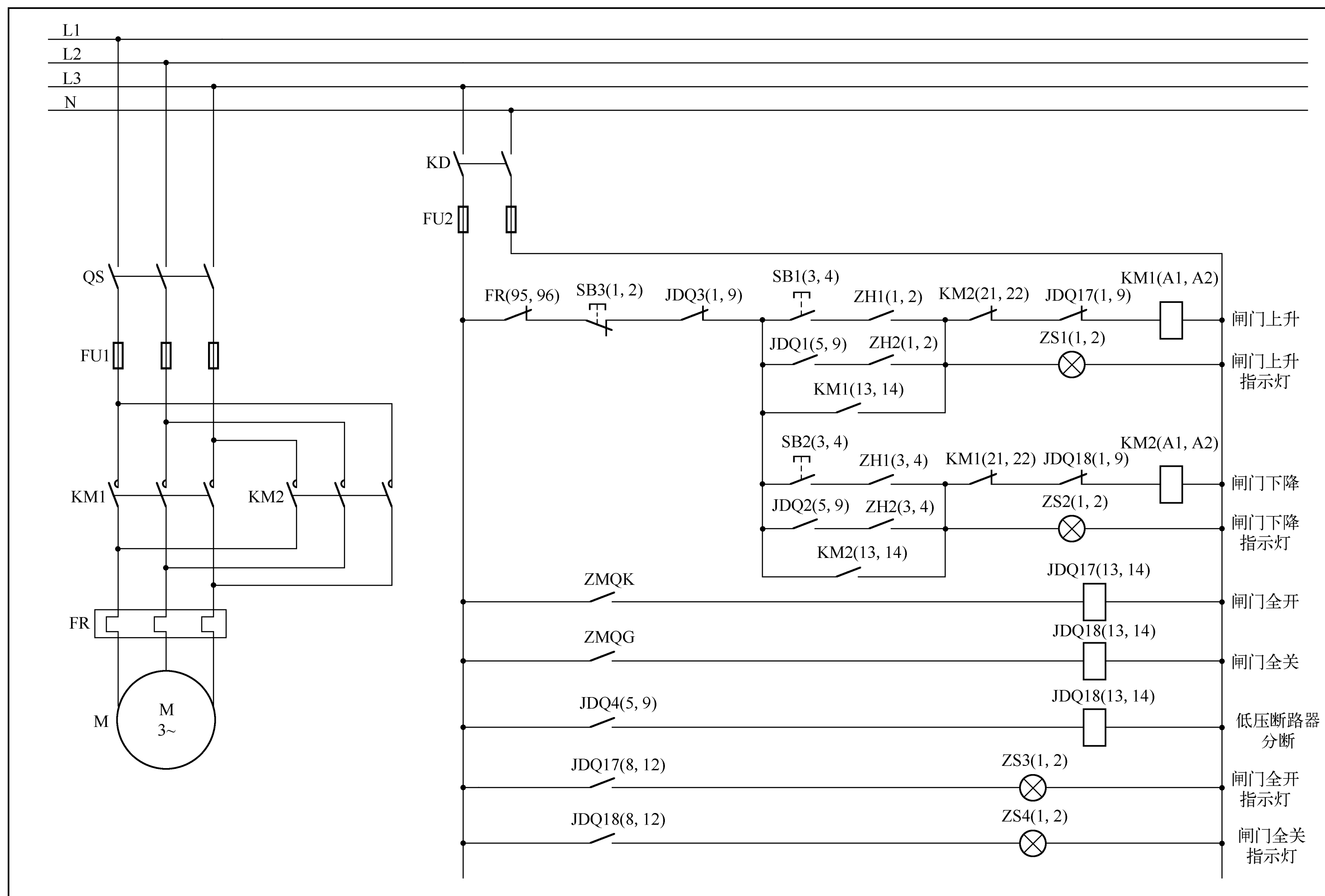



图 11–13　系统主回路及控制回路原理

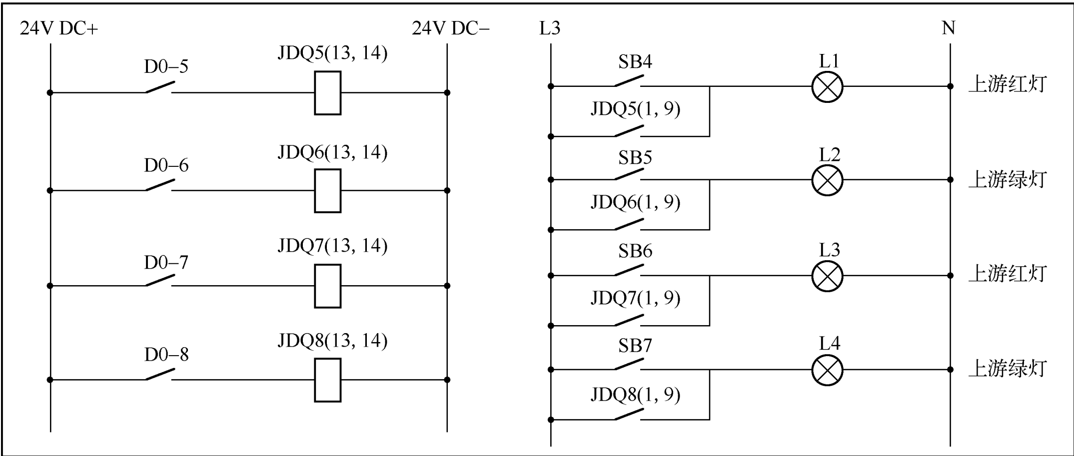



图 11−14　上下游通航灯控制原理图

**表 11−4　系统控制原理图各标记名与意义**

| 标　记　名 | 意　　义 | 标　记　名 | 意　　义 |
|---|---|---|---|
| L1、L2、L3 | 三相交流电 | KD | 控制回路电源空开 |
| N | AC 220V 零线 | ZH1 | 手动转换开关 |
| FU1、FU2 | 熔断器 | ZH2 | 手自动转换开关 |
| FR | 热继电器 | ZMQK | 闸门上限位 |
| QS | 低压断路器 | ZMQG | 闸门下限位 |
| M | 三相异步电动机 | SB1 | 停止按钮 |
| KM1、KM2 | 交流接触器 | SB2 | 上升按钮 |
| JDQ1 | 闸门上升继电器 | SB3 | 下降按钮 |
| JDQ2 | 闸门下降继电器 | SB4 | 上游通航红灯按钮 |
| JDQ3 | 闸门停止继电器 | SB5 | 上游通航绿灯按钮 |
| JDQ4 | 闸门主回路分断继电器 | SB6 | 下游通航红灯按钮 |
| JDQ5 | 上游通航红灯继电器 | SB7 | 下游通航绿灯按钮 |
| JDQ6 | 上游通航绿灯继电器 | L1 | 上游通航红灯 |
| JDQ7 | 下游通航红灯继电器 | L2 | 上游通航绿灯 |
| JDQ8 | 下游通航绿灯继电器 | L3 | 下游通航红灯 |
| JDQ17 | 闸门全开继电器 | L4 | 下游通航绿灯 |
| JDQ18 | 闸门全关继电器 | | |

本监控系统中，控制方式分为手动和自动两种独立的方式，手动控制方式在闸门启闭机站房的手动控制柜上操作，并且要在 PLC 不参与控制的情况下仍能正常的控制工作；自动控制方式是在管理所控制室的监控计算机上操作，利用组态软件向现地控制单元内的 PLC 发送控制命令，然后由 PLC 完成相应的控制任务。

在两种控制方式中，现场的手动控制为最高优先级。在现场手动控制方式时可以不受自动控制方式的干扰完成操作；可以拨动转换开关，将控制权限转为自动方式，由上位机进行操作控制；在上位机进行自动控制时，仍可以在现场随时干预控制过程。

在自动控制方式时，必须将现地控制单元的转换开关拨为自动方式，操作人员方能在控制室的监控计算机上对闸门的启闭、通航灯的亮灭等进行控制。

由系统控制原理图可设计出机柜内部接线表，如表 11-5、表 11-6、表 11-7 所示。

**表 11-5　PLC 机柜左端子接线表（PLC-L）**

| 端子号 | 左　标　记 | 右　标　记 | PLC 地址 | 信号意义 |
|---|---|---|---|---|
| 1 | DC 24V + | （端子 1-9 短接） | 1-9 端子短接 | DC 24V 开关电源正 |
| 2 | | DI-1、DI-21 | | 水位计回讯公共端 |
| 3 | | DO-1、DO-11 | | 闸门监控回讯公共端 |
| 4 | | AI-1 | | 通航红灯回讯公共端 |
| 5 | | | | |
| 6 | DC 24V - | （端子 11-19 短接） | 11-19 端子短接 | DC 24V 开关电源负 |
| 7 | | DI-20、DI-40 | | |
| 8 | | DO-10、DO-20 | | |
| 9 | | AI-20 | | |
| 10 | | | | |
| 11 | ZM1-1 | DI-2 | I0.0 | 水位计-1 |
| 12 | ZM1-2 | DI-3 | I0.1 | 水位计-2 |
| 13 | ZM1-3 | DI-4 | I0.2 | 水位计-3 |
| 14 | ZM1-4 | DI-5 | I0.3 | 水位计-4 |
| 15 | ZM1-5 | DI-6 | I0.4 | 水位计-5 |
| 16 | ZM1-6 | DI-7 | I0.5 | 水位计-6 |
| 17 | ZM1-7 | DI-8 | I0.6 | 水位计-7 |
| 18 | ZM1-8 | DI-9 | I0.7 | 水位计-8 |
| 19 | ZM1-9 | DI-12 | I1.0 | 水位计-9 |
| 20 | ZM1-10 | DI-13 | I1.1 | 水位计-10 |
| 21 | ZW1-11 | DI-14 | I1.2 | 水位计-11 |
| 22 | ZM1-12 | DI-15 | I1.3 | 水位计-12 |
| 23 | ZM1-13 | DI-16 | I1.4 | 闸门上升回讯 |
| 24 | ZM1-14 | DI-17 | I1.5 | 闸门下降回讯 |
| 25 | ZM1-15 | DI-18 | I1.6 | 闸门全开回讯 |
| 26 | ZM1-16 | DI-19 | I1.7 | 闸门全关回讯 |
| 27 | ZM1-17 | DI-22 | I2.0 | 闸门自动回讯 |
| 28 | ZM1-18 | DI-23 | I2.1 | 闸门手动回讯 |
| 29 | ZM1-19 | DI-24 | I2.2 | 上游通航红灯开回讯 |

（续）

| 端子号 | 左　标　记 | 右　标　记 | PLC 地址 | 信 号 意 义 |
|---|---|---|---|---|
| 30 | ZM1 – 20 | DI – 25 | I2. 3 | 上游通航绿灯开回讯 |
| 31 | ZM1 – 21 | DI – 26 | I2. 4 | 下游通航红灯开回讯 |
| 32 | ZM1 – 22 | DI – 27 | I2. 5 | 下游通航绿灯开回讯 |
| 33 | ZM1 – 23 | DI – 28 | I2. 6 | 主回路电源回讯 |
| 34 | ZM1 – 24 | DI – 29 | I2. 7 | 备用 |
| 35 | ZM1 – 25 | DI – 32 | I3. 0 | 备用 |
| 36 | ZM1 – 26 | DI – 33 | I3. 1 | 备用 |
| 37 | ZM1 – 27 | DI – 34 | I3. 2 | 备用 |
| 38 | | | | |
| 39 | YL1 – 2 | AI – 2 | IW4 | 闸室水位计 + |
| 40 | YL1 – 3 | AI – 3 | | 闸室水位计 – |
| 41 | YL2 – 1 | AI – 4 | IW6 | 闸门荷重计 + |
| 42 | YL2 – 2 | AI – 5 | | 闸门荷重计 – |
| 43 | YL3 – 1 | AI – 6 | IW8 | 主回路电压传感器 + |
| 44 | | AI – 7 | | 主回路电压传感器 – |
| 45 | | AI – 8 | IW10 | 主回路电流传感器 + |
| 46 | | AI – 9 | | 主回路电流传感器 – |
| 47 | | AI – 12 | IW12 | 备用 |
| 48 | | AI – 13 | | |

**表 11–6　PLC 机柜右端子接线表（PLC – R）**

| 端子号 | 左　标　记 | 右　标　记 | PLC 地址 | 信 号 意 义 |
|---|---|---|---|---|
| 1 | 220V – L | | 1 ~ 5 端子短接 | AC 220V 火线（进线） |
| 2 | DC 24V – L | | | DC 24V 开关电源 220V 火线 |
| 3 | PLC – L | | | PLC – PWR 火线 |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | 220V – N | | 7 ~ 11 端子短接 | AC 220V 零线（进线） |
| 8 | DC 24V – N | | | DC 24V 开关电源 220 V 零线 |
| 9 | PLC – N | | | PLC – PWR 零线 |
| 10 | | | | |
| 11 | | | | |
| 12 | | | | |

（续）

| 端子号 | 左 标 记 | 右 标 记 | PLC 地址 | 信 号 意 义 |
|---|---|---|---|---|
| 13 | JDQ1 －5 | ZMKZ －1 | Q0. 0 | 闸门上升控制（常开） |
| 14 | JDQ2 －5 | ZMKZ －2 | Q0. 1 | 闸门下降控制（常开） |
| 15 | JDQ3 －1 | ZMKZ －3 | Q0. 2 | 闸门停止（常闭） |
| 16 | JDQ1 －9 | ZMKZ －GG | | 闸门控制公共端（JDQ1 －9、JDQ2 －9、JDQ3 －9 短接） |
| 17 | JDQ4 －5 | ZMKZ －4 | Q0. 3 | 闸门主回路分断（常开） |
| 18 | JDQ4 －9 | ZMKZ －5 | | |
| 19 | JDQ5 －1 | ZMKZ －6 | Q0. 4 | 上游通航红灯控制（常闭） |
| 20 | JDQ5 －9 | ZMKZ －7 | | |
| 21 | JDQ6 －5 | ZMKZ －8 | Q0. 5 | 上游通航绿灯控制（常开） |
| 22 | JDQ6 －9 | ZMKZ －9 | | |
| 23 | JDQ7 －1 | ZMKZ －10 | Q0. 6 | 下游通航红灯控制（常闭） |
| 24 | JDQ7 －9 | ZMKZ －11 | | |
| 25 | JDQ8 －5 | ZMKZ －12 | Q0. 7 | 下游通航绿灯控制（常开） |
| 26 | JDQ8 －9 | ZMKZ －13 | | |
| 27 | JDQ9 －5 | ZMKZ －14 | Q1. 0 | 备用 |
| 28 | JDQ9 －9 | ZMKZ －15 | | |

**表 11–7　PLC 机柜继电器接线表**

| 继电器序号 | 左 黑 头 | PLC 端子号 | 信 号 内 容 |
|---|---|---|---|
| 1 | JDQ1 －13 | DO －2 | 闸门上升控制（常开） |
| 2 | JDQ2 －13 | DO －3 | 闸门下降控制（常开） |
| 3 | JDQ3 －13 | DO －4 | 闸门停止（常闭） |
| 4 | JDQ4 －13 | DO －5 | 闸门主回路分断（常开） |
| 5 | JDQ5 －13 | DO －6 | 上游通航红灯控制（常闭） |
| 6 | JDQ6 －13 | DO －7 | 上游通航绿灯控制（常开） |
| 7 | JDQ7 －13 | DO －8 | 下游通航红灯控制（常闭） |
| 8 | JDQ8 －13 | DO －9 | 下游通航绿灯控制（常开） |
| 9 | JDQ9 －13 | DO －12 | 备用 |
| 10 | JDQ10 －13 | DO －13 | 备用 |
| … | … | … | … |
| 17 | JDQ17 －13 | | 闸门全开中间继电器 |
| 18 | JDQ1813 | | 闸门全关中间继电器 |

在系统设计时为了减少信号干扰，并且更易于接线和系统的调试维护，通常将 PLC 机柜的左端子排用于接信号的输入端，右端子排用于接信号的输出端。

系统中使用的继电器型号为欧姆龙 MY2NJ 型继电器。该继电器的外观和引脚如图 11－15 所示。继电器的 13、14 脚连接的是继电器的线圈，为 DC 24V 线圈。在本系统中1～16 号继电器的 13 脚分别与 PLC 的 1～16 输出点相连接，用于 PLC 的输出控制。1～16 号继电器的 14 脚短接后，均接入左端子 DC 24V－中。继电器的 1、5、9 脚和 4、8、12 脚分别为一副 1 常开、1 常闭的触点。其中 9 脚和 12 脚各自为触点的公共端。本系统中的各个继电器只用到了 1 副触点，均使用的是 1、5、9 脚的触点。

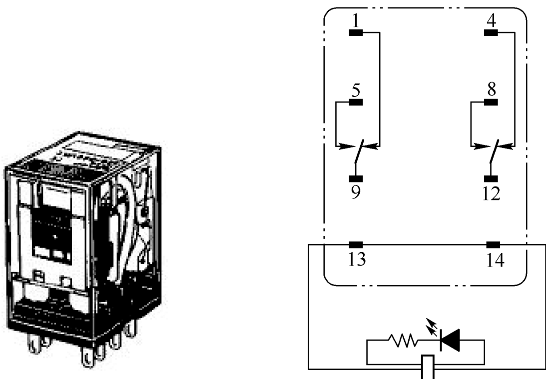



图 11－15　MY2NJ 型继电器的外观和引脚图

根据系统的控制原理图和 PLC 机柜的接线表可设计出如图 11－16 所示手动机柜内部设备布置图和如图 11－17 所示 PLC 机柜内部设备布置图。

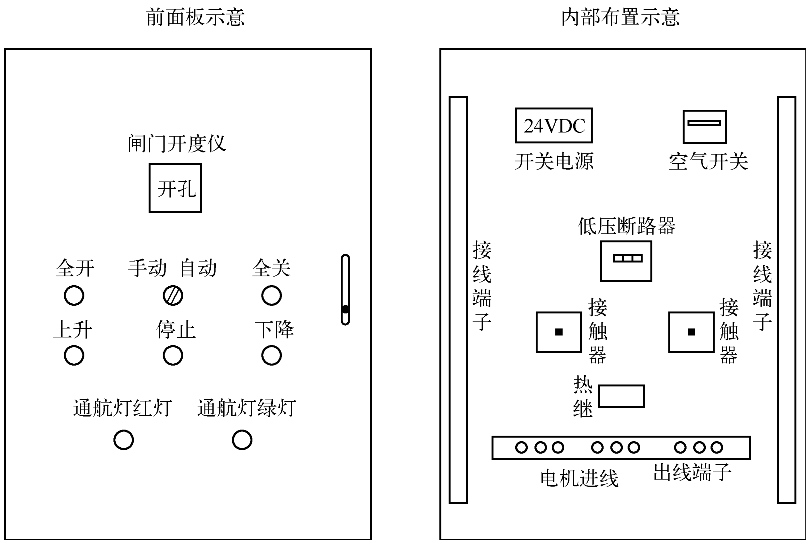



图 11－16　手动机柜内部设备布置示意图

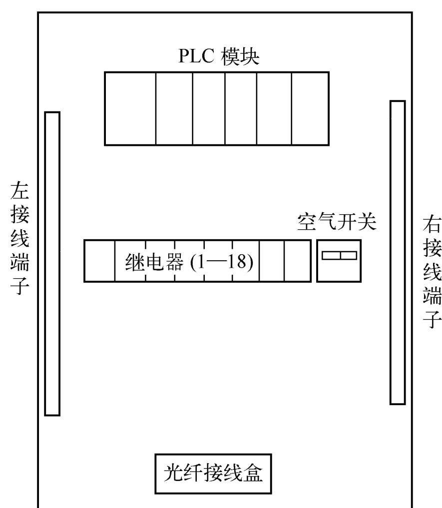



图 11-17 PLC 机柜内部设备布置示意图

### 11.4.5 设备组柜与接线工作

当系统控制原理图、机柜内部接线表和机柜内部设备布置图设计好后，就可根据图表进行组柜工作。在组柜接线时，应严格按照设计图进行，如遇到实际情况与设计不符时，应认真对比设计与实际相差的区别，找出原因，确定必须要更正的内容后尽快加以更正。

在接线时应认真、细致，尽量按接线表顺序接线，避免发生错误。在每根线的起始两端都做好标记。每根接线都要确保连接牢固，避免松脱、虚接。接好后要对每根接线进行检查并测量其通断情况及是否有短路情况。

### 11.4.6 PLC 硬件组态

编制好变量表后，就可以根据变量表并结合实际的控制要求编写程序。打开 STEP 7 后，首先建立一个新的项目，对新项目进行硬件组态，如图 11-18 所示。

对于硬件组态部分，此处需要进一步说明的是以太网通信模块 CP 343 -1 和 SSI 信号输入模块 SM 338 的硬件设置。

在图 11-18 中双击“CP 343 -1”模块，将出现 CP 343 -1 模块的属性设置对话框，如图 11-19 所示。CP 343 -1 模块的基本属性可使用默认值，需要更改的是模块的 IP 地址，单击“Properties”按钮，将出现 CP 343 -1 模块的 IP 地址设置对话框，如图 11-20 所示。将 CP 343 -1 模块的 IP 地址和子网掩码更改为所需要的内容，在本系统中没有使用网络路由，如系统使用了网络路由，还应在“Gateway”选项框中选择“Use router”，并设置相应的路由。

SM 338 是位置检测模块，每个 SM 338 模块具有 3 路输入，可以连接 3 路 SSI 信号的绝对值编码器输入。3 路绝对值编码器输入信号各自独立，可支持格雷码和二进制码数据格式。本系统使用的绝对值编码器输入信号为格雷码数据格式。

(0) UR

| | |
|---|---|
| 1 | PS 307 5A |
| 2 | CPU 314 |
| 3 | |
| 4 | CP 343-1 |
| 5 | DI32xDC24V |
| 6 | AI8x12Bit |
| 7 | DO16xDC24V/0.5A |
| 8 | SM 338 POS-INPUT |
| 9 | |
| 10 | |
| 11 | |

图 11-18　硬件组态图

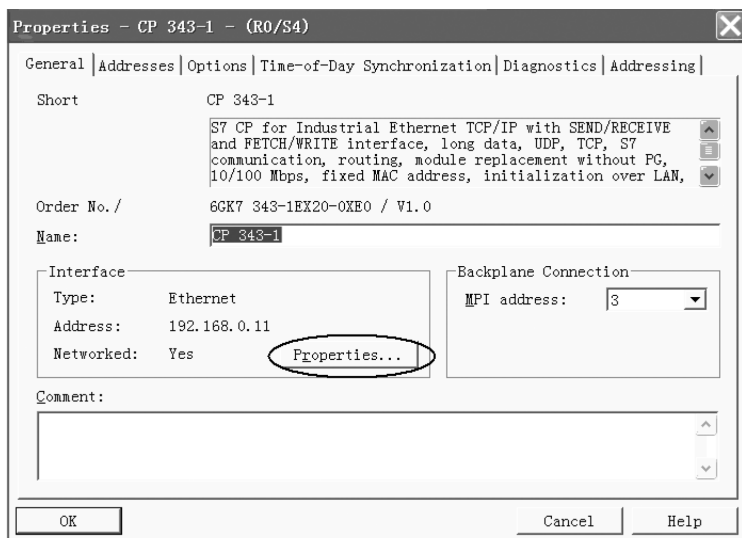



图 11-19　CP 343－1 属性设置对话框

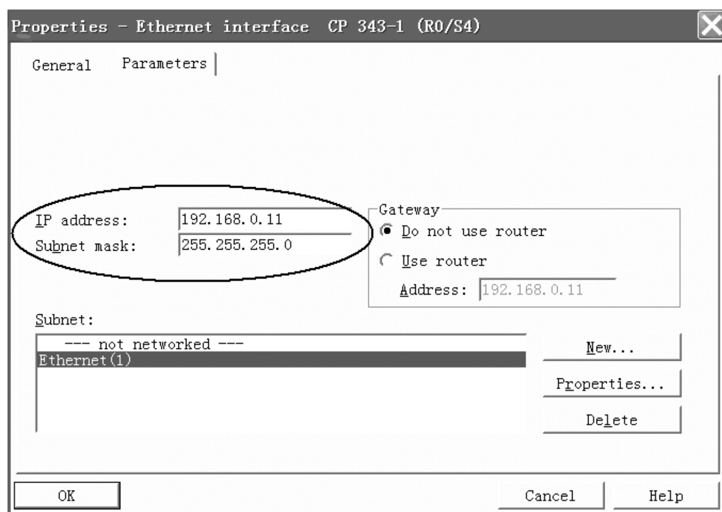



图 11-20　CP 343－1 IP 地址设置对话框

在图 11-18 中双击“SM 338”模块，将出现 SM 338 模块的属性设置对话框。单击“Addresses”选项卡，将出现 SM 338 模块的输入地址设置对话框，如图 11-21 所示。设置 SM 338 模块的输入点的起始地址。由于每路绝对值编码器输入的 SSI 信号为 25 位数据，所以 SM 338 的每路输入点数据为双字（DW）型。3 路 SSI 信号共占用 6 个字。

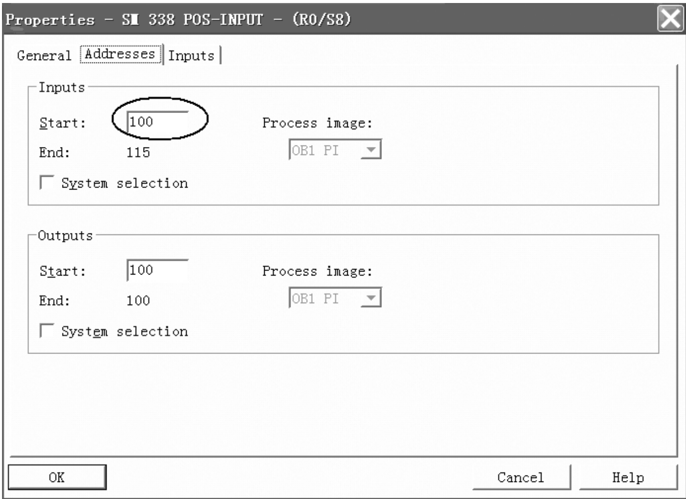



图 11-21　SM 338 Addresses 属性设置对话框

单击“Inputs”选项卡，将出现 SM 338 模块的输入通道属性设置对话框，如图 11-22 所示。设置 SM 338 模块相应各路通道的编码位数、编码类型、时钟频率等属性。

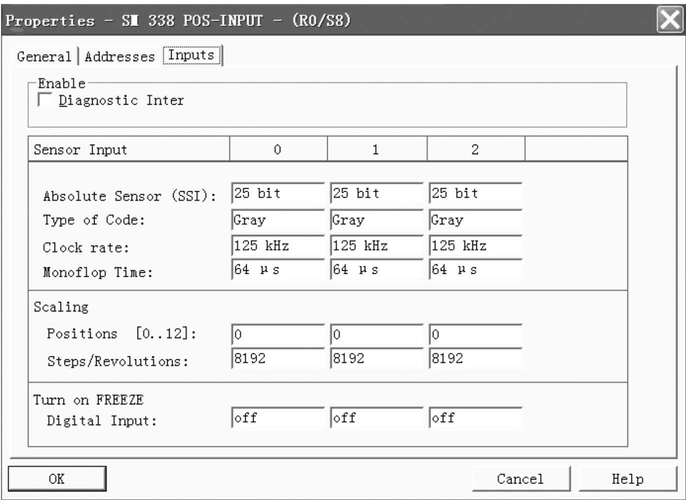



图 11-22　SM 338 模块的输入通道属性设置对话框

各模块的硬件属性设置完成后，单击“OK”按钮退出，并保存硬件组态。

## 11.4.7　软件编程设计与调试

在系统进行组柜、接线的同时，可进行 PLC 软件编程工作。编程前，首先要详细了解整个系统的控制内容、控制流程以及所要采集的各类信号。然后根据 PLC 接线表设计出 PLC 和上位机软件的变量表。本系统的 PLC 和上位机软件变量表如表 11-8 所示。

表 11-8　PLC 和上位机软件变量表

| I/O 地址 | PLC 与上位机地址 | 信号内容 | 类　　型 |
|---|---|---|---|
| I0.0 – I1.3 | M0.0 – M1.3 | 水位计 1–12 | PLC 回讯 |
| I1.4 | DB1.DBX2.0 | 闸门上升回讯 | |
| I1.5 | DB1.DBX2.1 | 闸门下降回讯 | |
| I1.6 | DB1.DBX2.2 | 闸门全开回讯 | |
| I1.7 | DB1.DBX2.3 | 闸门全关回讯 | |
| I2.0 | DB1.DBX2.4 | 闸门手动回讯 | |
| I2.1 | DB1.DBX2.5 | 闸门自动回讯 | |
| I2.2 | DB1.DBX2.6 | 上游通航红灯开回讯 | |
| I2.3 | DB1.DBX2.7 | 上游通航绿灯开回讯 | |
| I2.4 | DB1.DBX3.0 | 下游通航红灯开回讯 | |
| I2.5 | DB1.DBX3.1 | 下游通航绿灯开回讯 | |
| I2.6 | DB1.DBX3.2 | 主回路电源回讯 | |
| IW4 | DB1.DBW8 | 闸室内水位 | |
| IW6 | DB1.DBW12 | 闸门荷重 | |
| IW8 | DB1.DBW46 | 主回路电压 | |
| IW10 | DB1.DBW48 | 主回路电流 | |
| ID100 | DB1.DBD14 | 实时闸位值 | |
| | DB1.DBW10 | 闸室外水位 | |
| DB1.DBD22 | 预设闸门上限值 | 中间变量 | |
| | DB1.DBD18 | 预设闸门开高值 | 上位机命令 |
| | DB1.DBW42 | 闸室内水位 | |
| Q0.0 | DB1.DBX4.0 | 闸门上升控制 | |
| Q0.1 | DB1.DBX4.1 | 闸门下降控制 | |
| Q0.2 | DB1.DBX4.2 | 闸门停止 | |
| Q0.3 | DB1.DBX4.3 | 闸门主回路分断 | |
| Q0.4 | DB1.DBX4.4 | 上游通航红灯控制 | |
| Q0.5 | DB1.DBX4.5 | 上游通航绿灯控制 | |
| Q0.6 | DB1.DBX4.6 | 下游通航红灯控制 | |
| Q0.7 | DB1.DBX4.7 | 下游通航绿灯控制 | |
| | DB1.DBX5.0 | 闸位下限基值校准命令 | |
| | DB1.DBX5.1 | 上位机消警命令 | |

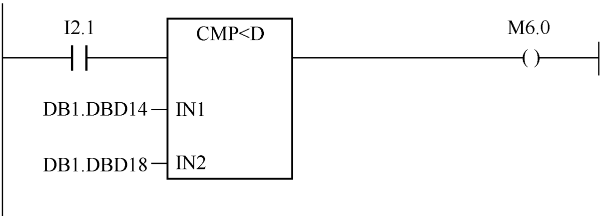
（续）

| I/O 地址 | PLC 与上位机地址 | 信 号 内 容 | 类　　型 |
|---|---|---|---|
| | DB1. DBX6. 0 | 闸门上升故障 | |
| | DB1. DBX6. 1 | 闸门下降故障 | |
| | DB1. DBX6. 2 | 闸门停止故障 | |
| | DB1. DBX6. 3 | 上游通航红灯控制故障 | 报警回讯 |
| | DB1. DBX6. 4 | 上游通航绿灯控制故障 | |
| | DB1. DBX6. 5 | 下游通航红灯控制故障 | |
| | DB1. DBX6. 6 | 下游通航绿灯控制故障 | |

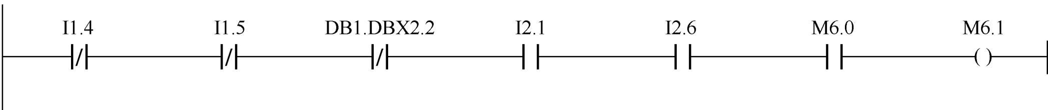
为了使程序有条理，清晰易懂，且便于调试，通常在编程序时应按各功能分段编程。例如，可以将程序分成信号采集部分、命令控制部分、报警判断部分等。然后再对各功能部分的程序细化。在编写程序时，应将各段程序做好相应的注释，以便程序调试和今后的系统维护。

图 11-23、图 11-24、图 11-25 分别为控制闸门上升、下降、停止的程序段。

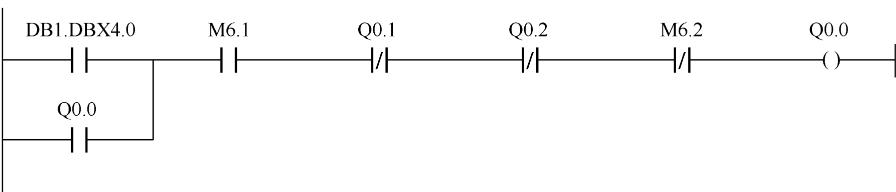
Network 38：闸门上升时，先判断当前闸门开高是否小于预设闸门开高




Network 39：闸门上升控制条件判断




Network 40：闸门上升控制




Network 41：延时500ms后释放上升继电器

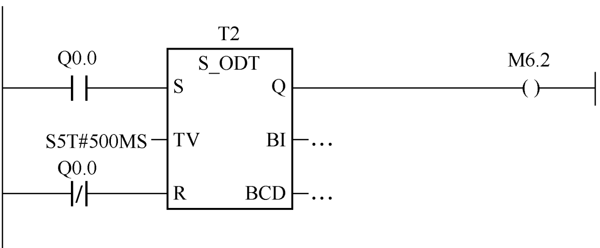
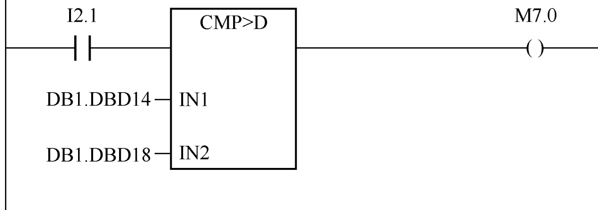
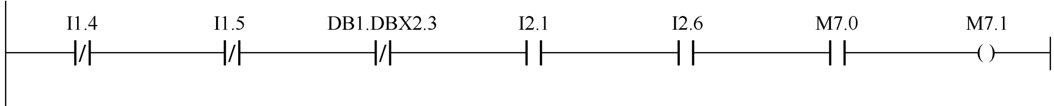



图 11-23　控制闸门上升的程序段

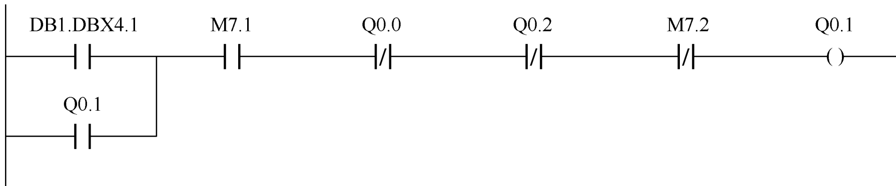
Network 42：闸门下降时，先判断当前闸门开高是否大于预设开高




Network 43：闸门下降控制条件判断




Network 44：闸门下降控制




Network 45：延时500ms后释放下降继电器

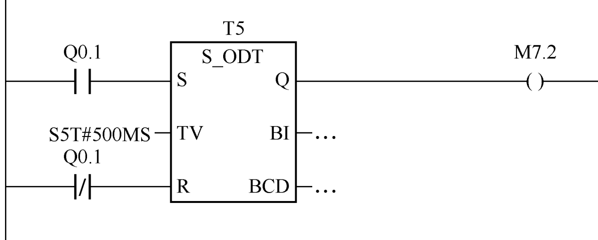


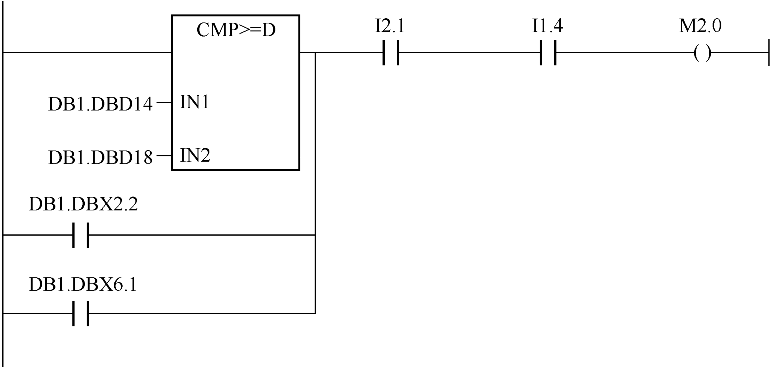

图 11-24　控制闸门下降的程序段

说明：图 11-23 中 Network 38 用来判断当前闸门的实时开高是否小于由上位机设定的预设闸门开高，如果小于，则允许闸门上升，否则上升的前提条件不满足。图中 Network 39 汇总了满足闸门上升的条件，对照表 11-8（PLC 和上位机软件变量表），可以很容易地明白 Network 39。只有闸门上升的全部条件满足，才能使 M 6.1 得电。图中 Network 40 中 Q0.1 和 Q0.2 是闸门上升的互锁条件，即在有 Q0.1 和 Q0.2 命令时不允许闸门上升。DB1.DBX4.0 为闸门上升的上位机命令，由图可以看出，当上位机发出闸门上升指令后，DB1.DBX4.0 得电，如果上升条件满足则此行网络接通，Q0.0 得电，并且将此命令自保持，以确保命令能被执行。M 6.2 用于在闸门上升命令下达后及时将命令释放掉，避免继电器长时间带电，是系统能够长期稳定运行的保护措施。

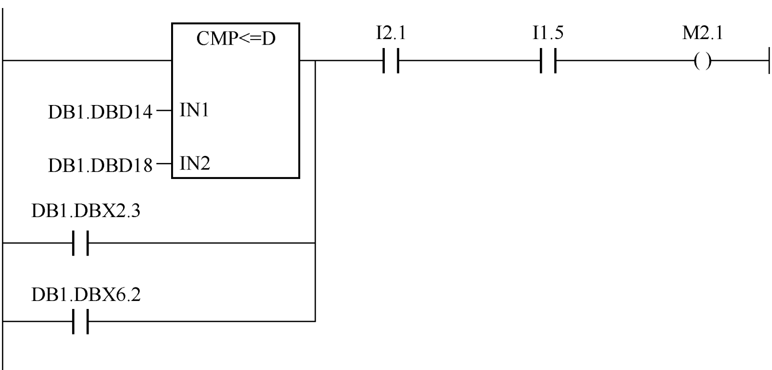
图 11-24 与图 11-23 的原理相似，这里不再赘述。

说明：图 11-25 是控制闸门停止的程序段，Network 55 是上升过程中满足停止的条件，由图可以看出共有 3 个条件，分别是闸位值到达预设开高、闸门开高到达上限和闸门上升故障。只要有一个条件满足，就可以使 M2.0 得电，引起 Q0.2 得电，使闸门停止运转。Network 56 与 Network 55 的原理相同。

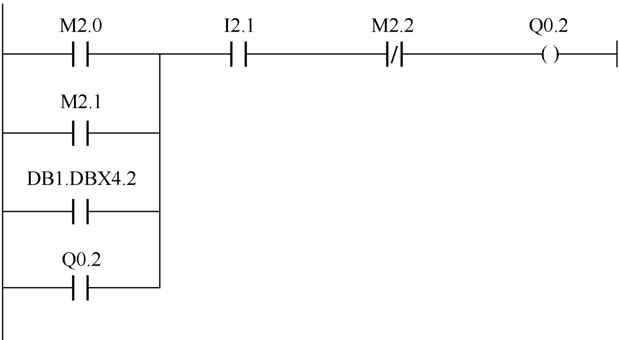
Network 55：闸门上升时停止条件




Network 56：闸门下降时停止条件




Network 57：闸门停止控制




Network 58：延时500ms后释放停止继电器

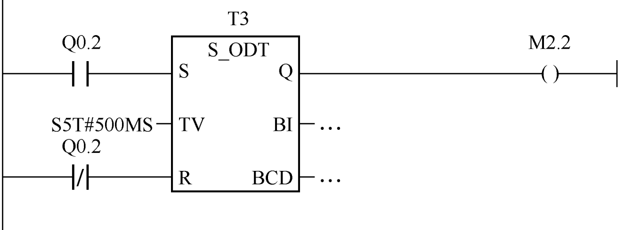



图 11-25　控制闸门停止的程序段

在程序编制好后，可以在计算机上利用 STEP 7 自带的 S7 - PLCSIM 对程序做仿真调试。在 STEP 7 窗口中的工具栏上点击“Simulation”按钮，打开 PLCSIM。利用 PLCSIM 的具体调试方法参见第 5 章。

在 PLCSIM 中对程序调试正常后，待系统的组柜和接线工作结束，可以将程序下载到 PLC 上做进一步的调试。从系统的安全角度考虑，第一次将程序下载到 PLC 上调试时应把系统的主回路 380 V 电源断掉，即做不带负载的调试。待调试的功能正常后，再做进一步的带负载的调试。

### 11.4.8 上位机软件设计

在 PLC 的编程调试过程中，可以同时进行系统的上位机监控软件的设计和编程工作。对上位机监控软件的设计编程主要使用组态软件完成。西门子公司为 S7 - 300 和 S7 - 400 系列 PLC 定制了自己的上位机组态软件 WinCC。利用 WinCC 可以很方便地完成监控画面的绘制、上位机组态、PC - PLC 通信等工作。这里对 WinCC 机组态软件不作详细的阐述，仅给出几幅监控画面，以说明监控系统的上位机操作。如图 11-26 所示为上位机监控主画面，如图 11-27 所示为上位机闸门监控画面，如图 11-28 所示为上位机单孔闸门监控画面，如图 11-29 所示为上位机数据查询画面。

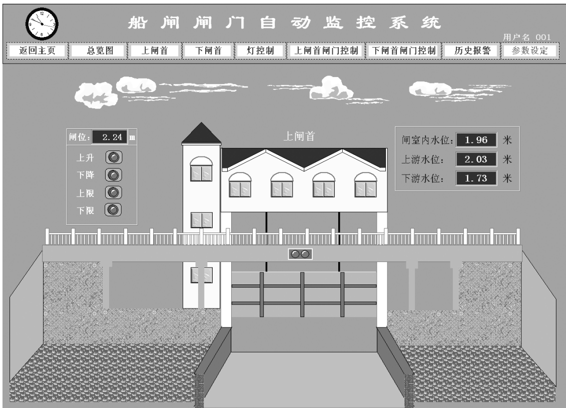



图 11-26　上位机监控主画面

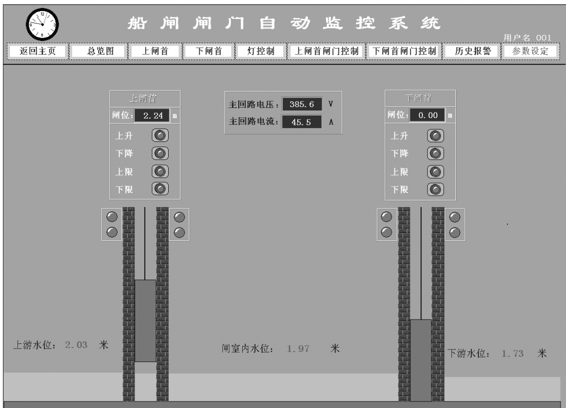



图 11-27　上位机闸门监控画面

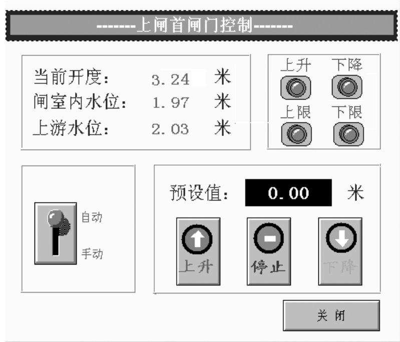



图 11-28　上位机单孔闸门监控画面

通州新江海河双桥套闸闸门自动监控

用户名 001

返回主页　总览图　上闸首　下闸首　灯控制　上闸首闸门控制　下闸首闸门控制　历史报警　参数设定

最新100条报警记录：

| 时间 | 状态 | 描述 | 组 | 操作员 | 操作员节点 |
|---|---|---|---|---|---|
| 11/28/2005 11:51:13 上午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 001 | JHW |
| 11/28/2005 11:54:03 上午 | ACK | 下闸首软启动故障 | zkalarm | | |
| 11/28/2005 11:54:03 上午 | ACK_RTN | 下闸首软启动故障 | zkalarm | | |
| 11/28/2005 12:39:45 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 001 | JHW |
| 11/28/2005 12:59:54 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 无 | JHW |
| 11/28/2005 01:00:43 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 无 | JHW |
| 11/28/2005 01:01:39 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 无 | JHW |
| 11/28/2005 01:01:56 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 无 | JHW |
| 11/28/2005 01:02:13 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 无 | JHW |
| 11/28/2005 01:02:33 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 无 | JHW |
| 11/28/2005 01:03:51 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 无 | JHW |
| 11/28/2005 01:05:05 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 无 | JHW |
| 11/28/2005 01:11:11 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 无 | JHW |
| 11/28/2005 02:06:47 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 无 | JHW |
| 11/28/2005 02:43:20 下午 | ACK_ALM | GGG | zkalarm | 001 | JHW |
| 11/28/2005 02:54:50 下午 | ACK_RTN | GGG | zkalarm | 001 | JHW |
| 11/28/2005 03:18:55 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 001 | JHW |
| 11/28/2005 03:19:41 下午 | ACK | 下闸首软启动故障 | zkalarm | | |
| 11/28/2005 03:19:41 下午 | ACK_RTN | 下闸首软启动故障 | zkalarm | | |
| 11/28/2005 03:20:48 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 001 | JHW |
| 11/28/2005 03:22:30 下午 | ACK | 下闸首软启动故障 | zkalarm | | |
| 11/28/2005 03:22:30 下午 | ACK_RTN | 下闸首软启动故障 | zkalarm | | |
| 11/28/2005 03:45:23 下午 | UNACK_ALM | 下闸首软启动故障 | zkalarm | 001 | JHW |
| 11/28/2005 03:45:37 下午 | ACK | 下闸首软启动故障 | zkalarm | | |
| 11/28/2005 03:45:37 下午 | ACK_RTN | 下闸首软启动故障 | zkalarm | | |

图 11-29　上位机数据查询画面

### 11.4.9　系统联调

当 PLC 的软件调试均正常且上位机监控软件编制结束后，接下来要进行系统的联合调试。首先，在实验室按照正常的控制步骤分阶段的将各个控制步骤逐一进行测试，确保每个环节正确无误。然后进行整个控制过程的连贯运行测试。当一切测试正常后，再进行模拟故障报警情况的调试。

当在实验室所有测试均正常后，该系统的硬件故障、软件程序和逻辑问题已基本排除，然后就可以到项目现场进行安装和调试。在现场调试过程中，会遇到实验室中不可预料的软硬件问题，这时应冷静分析找到解决办法，直到完全符合控制要求为止。

## 11.5　某钢厂大电炉水处理自动化监控系统工程实例

### 11.5.1　项目概况和要求

某钢厂大电炉水处理自动化监控系统工程项目的实施，可实现整个大电炉水处理自动化系统工程的现地和远方控制操作、主要参数的实时监测、故障报警、运行过程模拟显示等功

能。另外还可以有效地提高系统设备的可靠性、安全性，以及控制操作的自动化水平。能够实时对设备参数和运行工况进行监视，消除设备运行隐患，确保主设备的完好率和可调率，减轻运行人员劳动强度。

本实例所设计的某钢厂大电炉水处理自动化监控系统监控范围为：循环水泵房（主要包括加热炉净循环水泵房、轧钢电机净循环水泵房、浊循环水泵房等）、一次、二次沉淀池及阀门间过滤器。系统监控的内容主要包括所有泵的启/停和所有电磁阀的开/关。系统具有手动和自动两种操作模式，并能根据所监控设备的状态、压力、液位、温度和流量等实时情况进行报警和应对修正控制，具有安全联锁等功能。

## 11.5.2 系统总体设计

根据要求，本系统配置了两台工控机作为上位机，对水处理系统设备进行监视和控制。为增加系统的可靠性，两台上位机采用双机冗余热备方式。下位机对现场各种检测仪表和电机设备等进行信号采集与控制。两者之间采用以太网络高速通信，完成各种数据的交换。

上位机由两台监控主机组成，采用 WinCC 组态软件进行上位机监控系统开发。两台主机互为热备，每台主机均可以独立对钢厂大电炉水处理系统进行监控。并配备以太网通信接口，与 PLC 系统实现高速通信。

下位机控制系统以西门子 S7－300 系列 PLC 为核心，并通过 PROFIBUS 网络与其他站点完成通信。水处理系统的各种仪表采集信号，可以方便地与主 CPU 通信，并具有灵活的扩展功能。

如图 11－30 所示为该系统的拓扑图，循环水泵房现地 LCU 主要由一套 PLC 和一套 ET200M 组成，分别作为主站和 1 号从站。监控循环泵房内的设备包括水泵、阀门、冷却塔电机，以及采集加热炉净环水冷/热水池、轧机净环水冷/热水池和浊环水水池的液位、水流量、水压，监控进出冷却塔水温等。

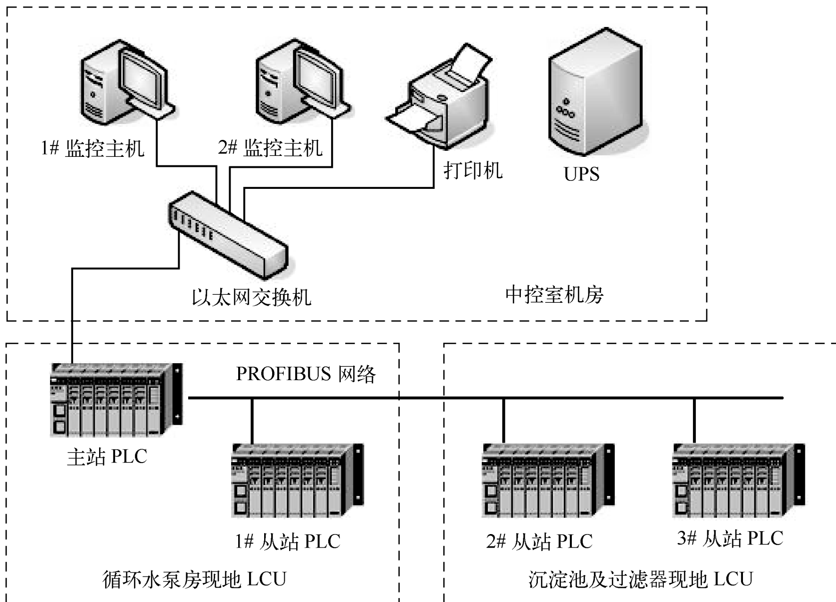



图 11－30　系统的拓扑图

沉淀池及过滤器现地 LCU 主要由两套 ET200M 组成，作为 2 号和 3 号从站。其包括监控水泵、阀门、阀门间过滤器三通阀等设备，以及显示一次沉淀池、二次沉淀池的液位、流量、水压和过滤器水流量、压差等。

### 11.5.3 PLC 模块及其他设备的选型

1. 系统点数统计

（1）数字量输出

在本系统中，主要控制对象为循环水泵房、二次沉淀池及阀门间过滤器。具体的 DI/DO 点数量与泵和阀相关。经统计得知，本系统的泵有 30 个，阀有 20 个。

对于每个泵，控制方式有起动和紧急停机，需要两个输出点；对于每个阀，控制方式有开阀和关阀（即：正转和反转），也需要两个输出点，如表 11-9 所示。

表 11-9 PLC 变量表

| I/O 地址 | 信号内容 | 类型 |
|---|---|---|
| Q4.0 | 1#101M 泵起动 | PLC 命令 |
| Q4.1 | 1#101M 泵紧急停机 | |
| Q10.4 | 1#501VM 阀正转 | |
| Q10.5 | 1#501VM 阀反转 | |
| … | … | |
| I 0.0 | 1#101M 泵集中控制 | PLC 回讯 |
| I 0.1 | 1#101M：泵准备好 | |
| I 0.2 | 1#101M：泵运行 | |
| I 0.3 | 1#101M：泵故障 | |
| I 5.0 | 1#501VM SQ1 阀全开 | |
| I 5.1 | 1#501VM SQ2 阀全关 | |
| I 5.2 | 1#501VM SQ3 阀开过力矩 | |
| I 5.3 | 1#501VM SQ4 阀关过力矩 | |
| … | … | |

（2）数字量输入

PLC 回讯信号也就是输入点对于泵和阀都需要 4 个点，信号内容同样见表 11-9。

（3）模拟量输入

系统要采集水处理各环节中水的液位、流量、水压和水温，需要多路模拟量输入通道。经分析统计，本系统要采集液位 14 个、流量 14 个、水压 14 个、温度 7 个。

2. PLC 的选型

本系统选用一个 CPU 315DP 为控制核心，由于 DI/DO 点很多，超出了一个机架的最大模块数量，所以用 ET200M 扩展 I/O 点。也可以采用多 CPU 的方式，但是系统造价会随之升高。根据以上分析，系统共需要扩展 3 套 ET200M，如图 11-29 所示，循环水泵房现地 LCU 主要由一套 PLC 和一套 ET200M 组成，分别作为主站和 1 号从站，沉淀池及过滤器现

地 LCU 主要由两套 ET200M 组成，作为 2 号和 3 号从站，主站与 3 个从站通过 PROFIBUS 网络系统连接。

主站的硬件配置如图 11-31 所示，除了 CPU 315 -2DP，还组态了以太网模块 CP 343 -1，用于与上位机 WinCC 的通信；另外根据需要组态了 AI 和 DO 模块。温度采集需要用模拟量模块的 RTD（热电阻）模块。

(0) UR

| | |
|---|---|
| 1 | PS 307 10A |
| **2** | **CPU 315-2 DP** |
| *X2* | *DP* |
| 3 | |
| 4 | CP 343-1 |
| 5 | AI8x16Bit |
| 6 | AI8x16Bit |
| 7 | AI8x16Bit |
| 8 | AI8xRTD |
| 9 | DO32xDC24V/0.5A |
| 10 | |

图 11-31　主站的硬件配置

1 号从站由扩展的一套 ET200M 承担，硬件配置如图 11-32 所示，根据需要配置多个 DI 和 DO 模块。2 号和 3 号从站的配置与主站和 1 号从站的配置相似，较为简单，在此不再赘述。

(3)　IM 153-1

| 插... | 模块 ... | 订货号 ... | I 地址 | Q 地址 | 注.. |
|---|---|---|---|---|---|
| 1 | | | | | |
| *2* | *IM 153-1* | *6ES7 153-1AA03-0XB0* | *2045* | | |
| 3 | | | | | |
| 4 | DI32xDC24V | 6ES7 321-1BL00-0AA0 | 0...3 | | |
| 5 | DI32xDC24V | 6ES7 321-1BL00-0AA0 | 4...7 | | |
| 6 | DI32xDC24V | 6ES7 321-1BL00-0AA0 | 8...11 | | |
| 7 | DO32xDC24V/0.5A | 6ES7 322-1BL00-0AA0 | | 4...7 | |
| 8 | DI32xDC24V | 6ES7 321-1BL00-0AA0 | 12...15 | | |
| 9 | DI32xDC24V | 6ES7 321-1BL00-0AA0 | 16...19 | | |
| 10 | DO32xDC24V/0.5A | 6ES7 322-1BL00-0AA0 | | 8...11 | |
| 11 | | | | | |

图 11-32　1 号从站的硬件配置

### 11.5.4　控制原理图及设备接线图的设计

控制原理图与 11.4 节相似，此处不再赘述。

本系统配置了两个机柜，一个是循环水泵房 PLC 机柜，另一个是沉淀池及过滤池 PLC 机柜。如图 11-33 所示为循环水泵房 PLC 机柜设备布置图，接线端子排布置于机柜的背面。沉淀池及过滤池 PLC 机柜布置图与循环水泵房 PLC 机柜相似，此处不再赘述。

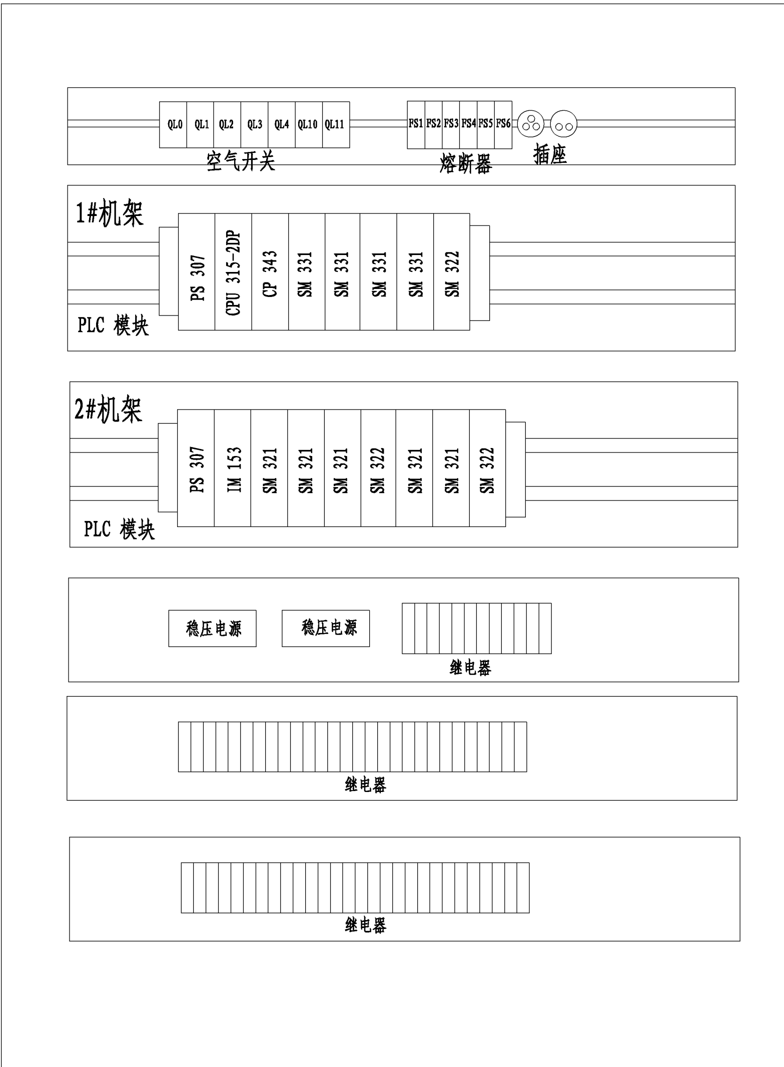



图 11-33　循环水泵房 PLC 机柜设备布置图

本系统中配置了多个 AI 模块、DI 模块和 DO 模块，作为示例，仅在此列举出其接线图各一幅，如图 11-34、图 11-35、图 11-36 所示。

### 11.5.5　设备组柜与接线工作

当系统控制原理图、机柜内部接线表和机柜内部设备布置图设计好后，就可根据图表进行组柜工作。在组柜接线时，应严格按照设计图进行，如遇到实际情况与设计不符时，应认真对比设计与实际相差的区别，找出原因，确定必须要更正的内容后尽快加以更正。

在接线时应认真、细致，尽量按接线表顺序接线，避免发生错误。在每根线的起始两端都做好标记。每根接线都要确保连接牢固，避免松脱、虚接。接好后要对每根接线进行检查并测量其通断情况及是否有短路情况。

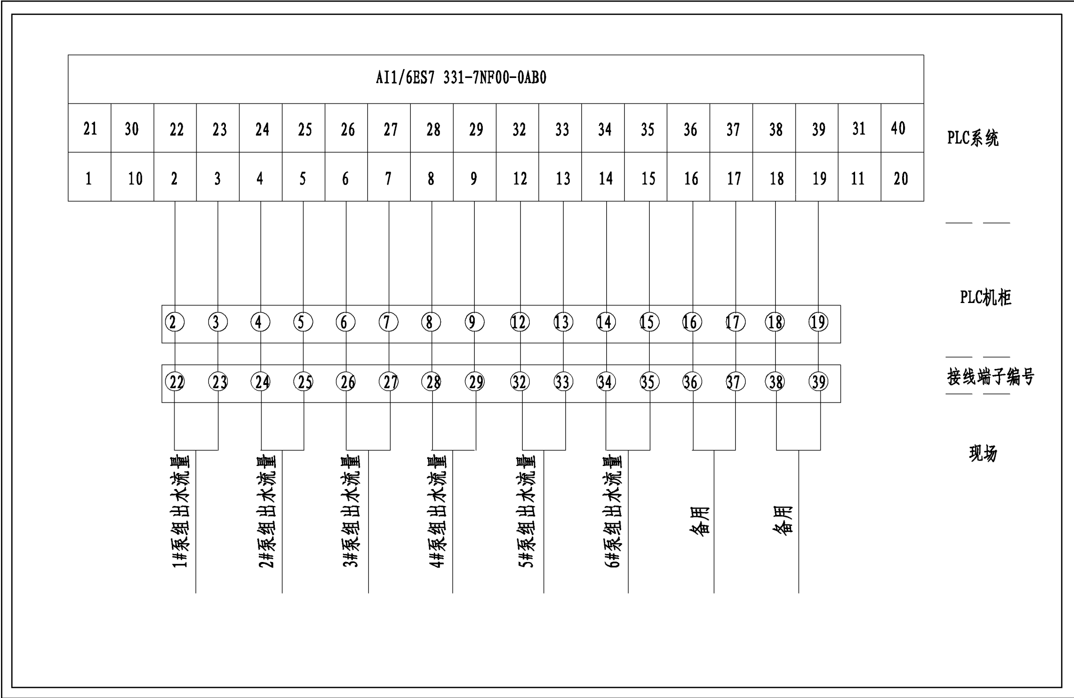



图 11-34　某 AI 模块接线图

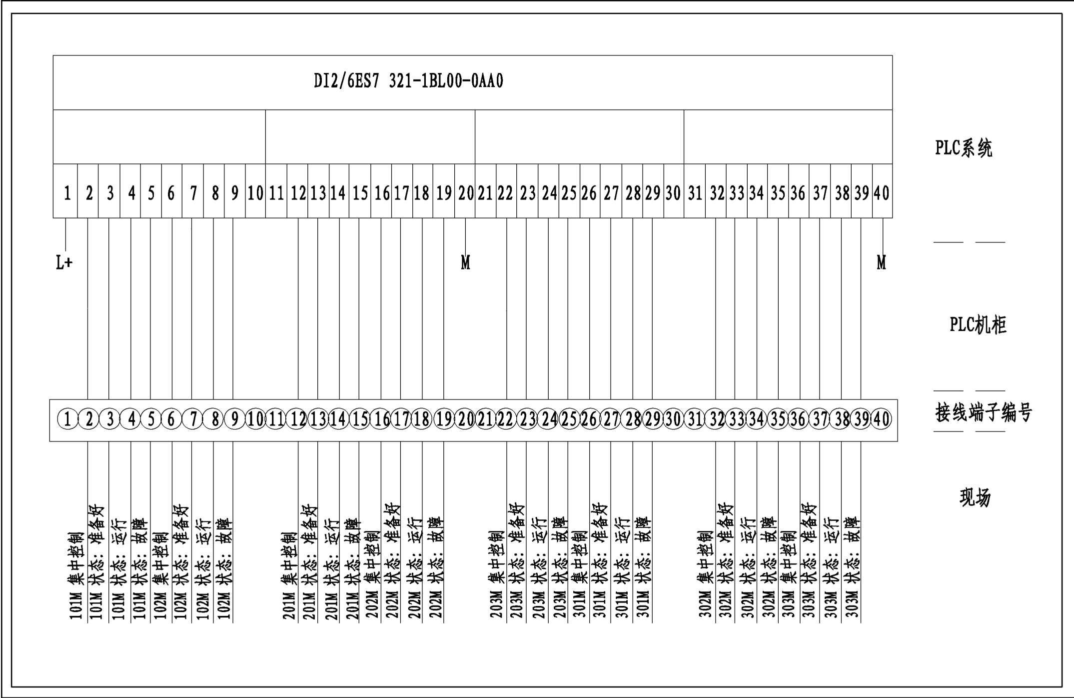



图 11-35　某 DI 模块接线图

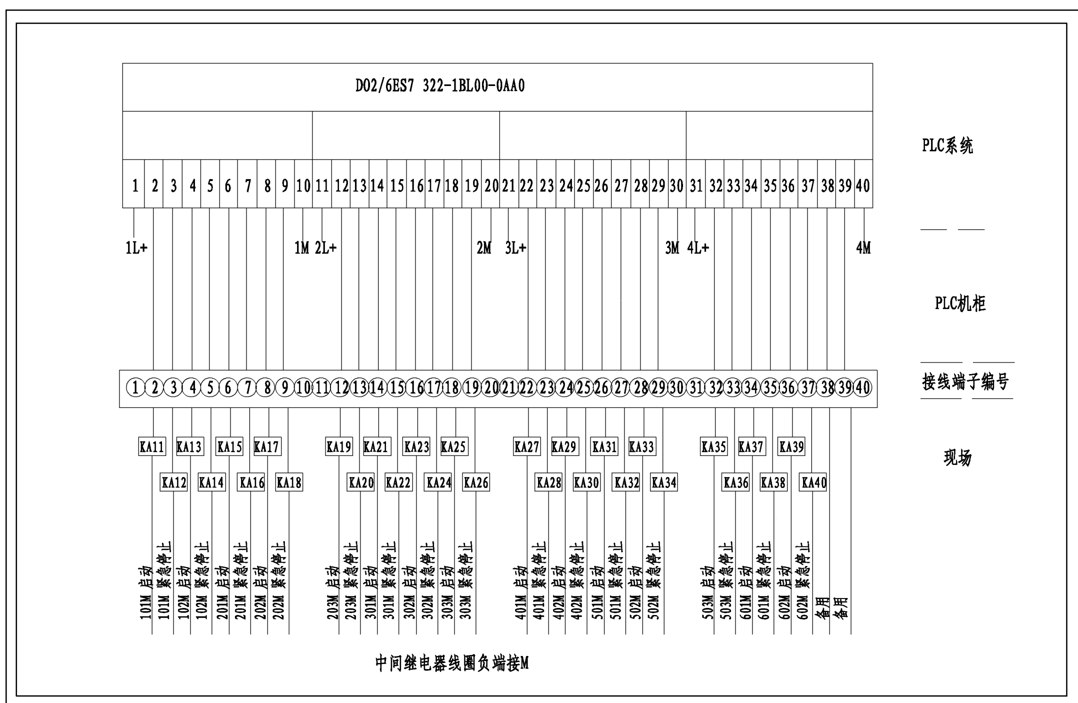



图 11-36　某 DO 模块接线图

## 11.5.6　PLC 硬件组态

如图 11-30 所示，本工程涉及两个网络，一个是 CPU 315 与上位机 WinCC 的以太网，另一个是 CPU 315 与 3 个 ET200M 的 PROFIBUS 通信网络。以太网是实现 PLC 系统与上位机之间的通信，通过在主站中组态的以太网模块 CP 343 - 1 完成；PROFIBUS 通信将在此处详述。

首先保证物理连接，用 PROFIBUS 电缆和接头将 CPU 315 与 3 个 ET200M 连接。并在 3 个 ET200M 硬件上分别设置拨码数字（即：其 PROFIBUS 地址）为 3、4 和 5。

新建一个项目后，建立 PROFIBUS 网络，将 CPU 315 设置为主站，PROFIBUS 地址为 2，并组态主站机架上的其他模块，硬件组态图如图 11-37 所示，以太网模块 CP 343 - 1 的组态可参考 11.4 节。

这里详细介绍 ET200M 的组态。在图 11-37 的右侧硬件目录中选择“PROFIBUS DP/ET200M/IM 153 - 1”，按住此处并把它拖到 PROFIBUS 电缆外。根据提示可组态为从站，并设置每个 ET200M 的 PROFIBUS 地址，图 11 - 37 中是组态好的 PROFIBUS 地址为 3 的 ET200M。注意一定要与硬件拨码上的地址一致，而且保证不能和其他站点地址冲突。单击选择挂在 PROFIBUS 电缆下的 ET200M，可以选择和组态其扩展的 I/O 模块，需要在刚刚的目录下选用，方法与基本的硬件组态相同。

这样即完成该项目完整的硬件组态，如图 11-38 所示。主站的 PROFIBUS 地址为 2，三个从站的地址分别为 3、4 和 5。

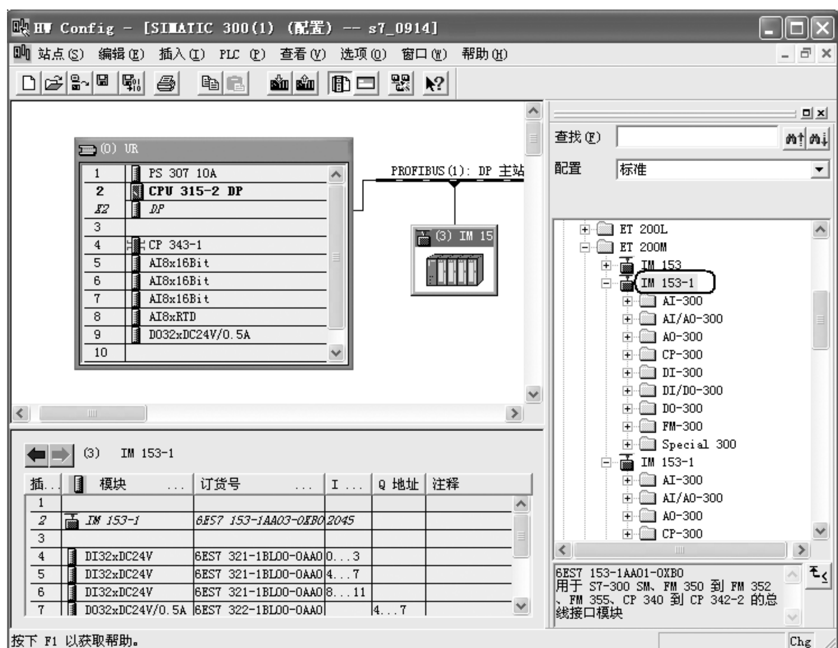



图 11-37　主站硬件组态图和组态 ET200M

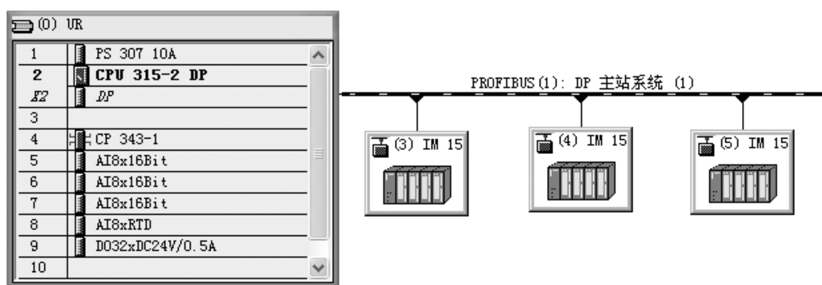



图 11-38　完整的硬件组态图

## 11.5.7　软件编程设计与调试

在系统进行组柜、接线的同时，可进行 PLC 软件编程工作。编程前，首先要详细了解整个系统的控制内容、控制流程以及所要采集的各类信号。然后根据 PLC 接线表设计出 PLC 和上位机软件的变量表。

PLC 主程序 OB1 是主循环程序，由多个子程序组成，以下将介绍其中 4 个主要子程序的编程。

### 1. 模拟量数据采集子程序

本项目中模拟量包括液位、流量、水压和水温等，需要多路模拟量输入通道。在硬件上配置了多个 SM331 模块，用于完成各个模拟量数据的采集、上位机显示、记录与报警等功能。

在模拟量处理过程中，多数采集液位、流量和水压的传感器的输出信号为 4～20mA 的电流信号。对于这类信号，软件可以用 STEP 7 系统中自带的 FC105 子程序完成。在目录

"Standard Library/TI－S7 Converting Blocks" 下可以找到 FC105，其功能是完成传感器输入值的线性转换。

如图 11－39 所示为流量值的线性转换，PIW256 是传感器输入，并由 AI 模块转换为数字量，PIW256 的范围为 0～27648；转换的上限和下限分别为 600 和 0，M10.0 初始为 0，代表单极性，OUT 输出为实型，存于 DB1 数据块中，供上位机使用。

对于温度采集，由于使用的是 RTD 模拟量输入模块，与普通模拟量的处理方式不同。处理子程序如图 11－40 所示，在 FC1 中调用它，参数 "IN" 为传感器输入值，除 10 后，转换为实数类型的温度值给参数 "OUT"，输出也存于 DB1 数据块中，供上位机使用。这个转换的操作也可以在上位机中进行。

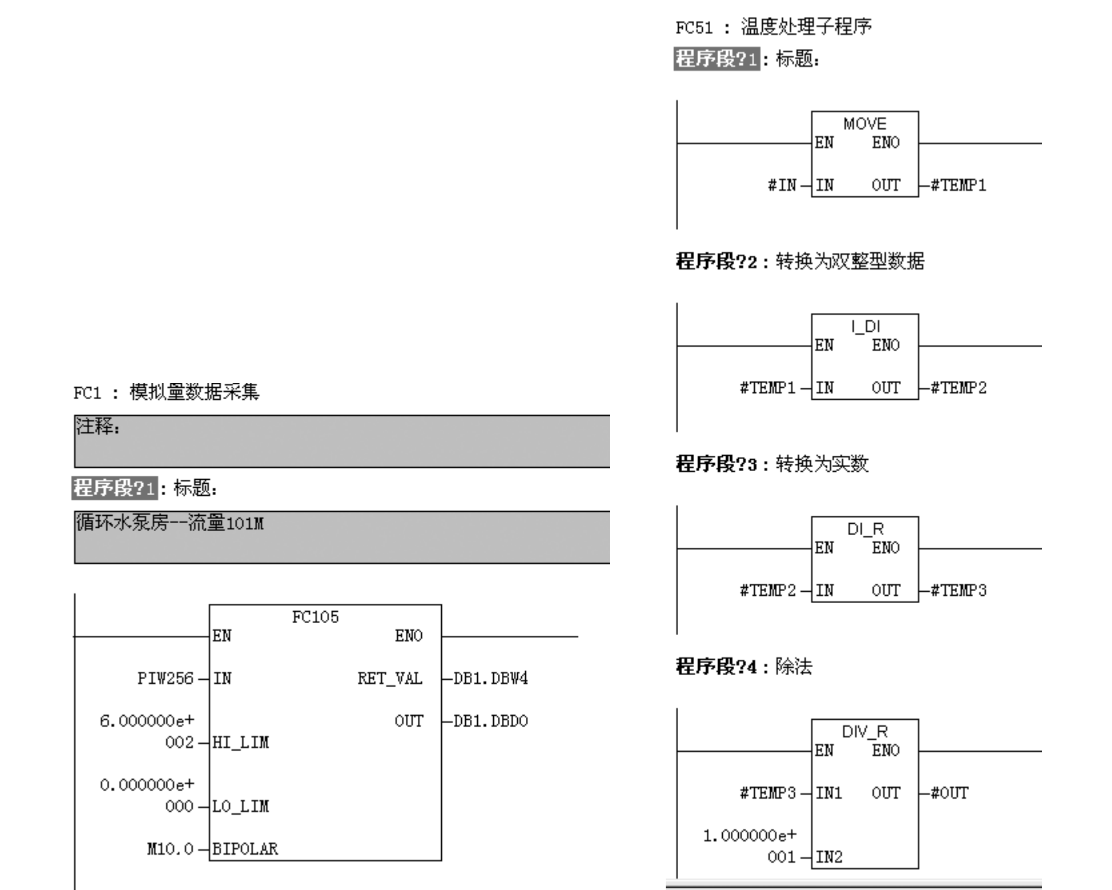



图 11－39　流量处理子程序　　　　图 11－40　温度处理子程序

### 2. 报警子程序

本系统在正常工作过程中，水位、温度和管道压力均有其正常的阈值，当 PLC 采集到超过或低于正常范围值的水位、温度或管道压力时，应产生相应的报警，供工作人员查询。如图 11－41 所示为水位过高、过低报警子程序。

### 3. 主站泵起动停止子程序

本系统中对泵的控制分为两种类型，一类是水泵管路中不带蝶阀的，另一类是水泵管路中串有蝶阀的。两类泵的控制方式有所不同。本子程序处理针对前者。

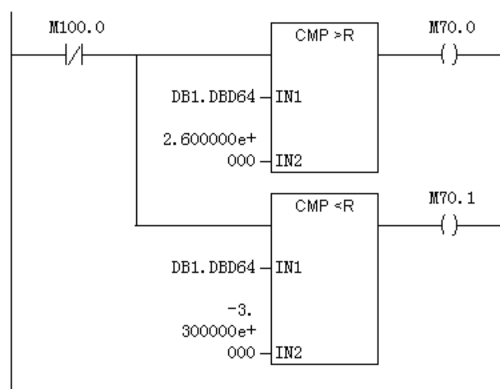



图 11-41　部分报警子程序

当现地控制柜转换开关置于“集中控制”状态时，允许上位机进行远程集中控制。以主站 101M 水泵为例，当 101M 水泵回讯信号正常时，上位机按下“起动”按钮，将控制 101M 水泵起动。另外，本系统在“集中控制”状态下，101M 与 102M 水泵之间又做了互为备用的联锁控制，即 101M 与 102M 水泵均可单独启停，在 102M 水泵运行过程中出现故障且当“1#1 组故障自投介入”按钮处于按下状态时，101M 水泵将自动起动运行。程序如图 11-42 所示。

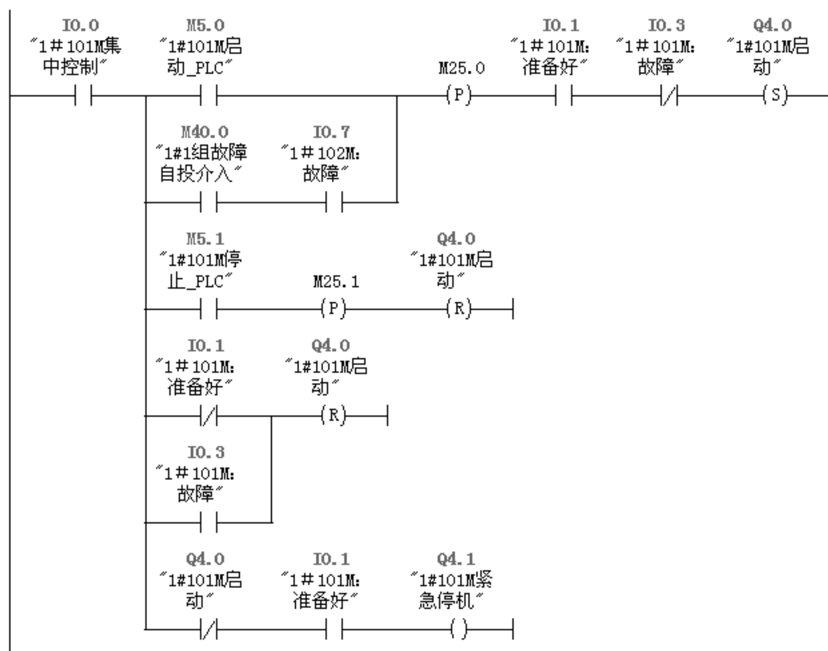



图 11-42　主站泵控制子程序

#### 4. 主站泵阀起动停止子程序

本子程序处理在回路中既有泵又有蝶阀的情况，所以程序相比上一类子程序会复杂些。

本系统中要求主站 601M 的泵和蝶阀在起动和停止控制过程中需要相互的联锁控制，具体的泵阀起动和停止控制流程图分别如图 11-43 和图 11-44 所示。根据泵阀控制的流程要求，编制 PLC 控制程序，如图 11-45 所示。

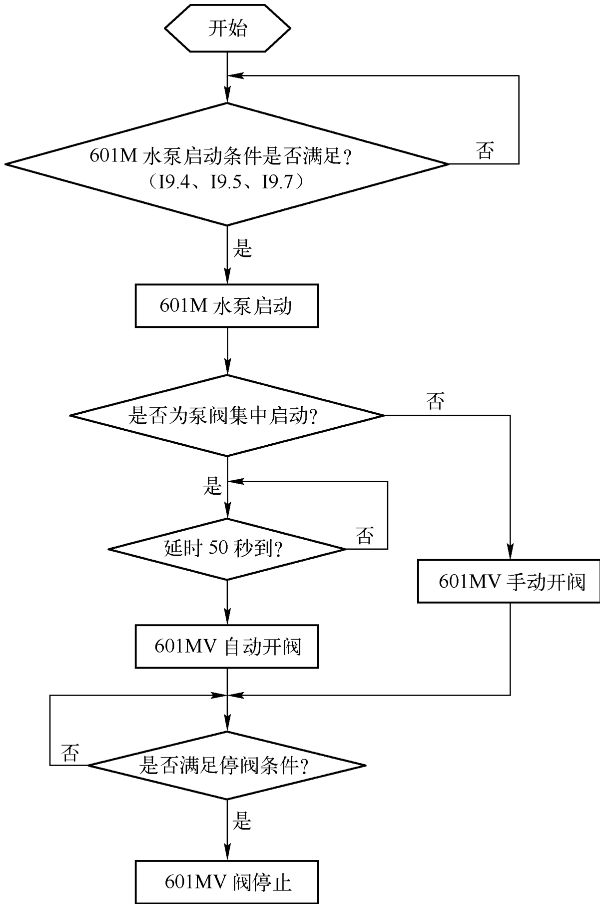



图 11-43　主站 601M 泵阀起动流程图

## 11.5.8　上位机软件设计

本系统使用 WinCC 软件设计上位机的监控界面，根据项目设计要求，先设计上位机与用户进行交互的界面总体框架图。如图 11-46 所示为界面总体框架图。

如图 11-47 所示为循环水泵房的检测控制画面，如图 11-48 所示为循环水泵房检测控制流程图中的部分泵阀控制界面。

## 11.5.9　系统联调

参见 11.4.9 节。

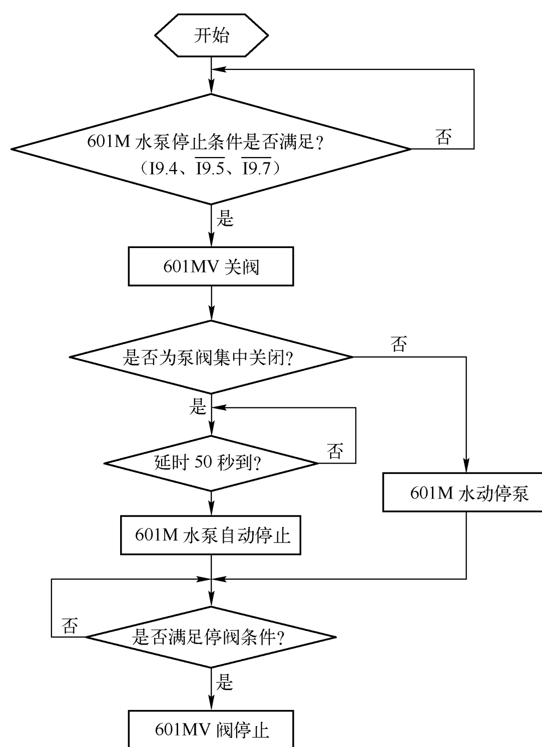
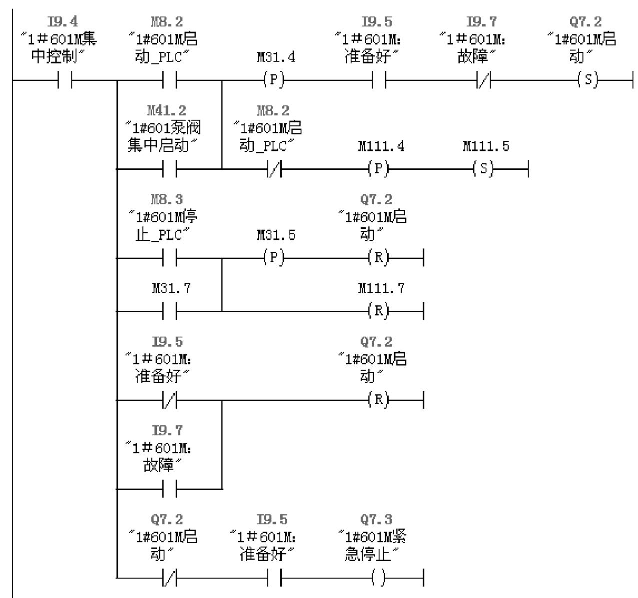



图 11-44　主站 601M 泵阀停止流程图

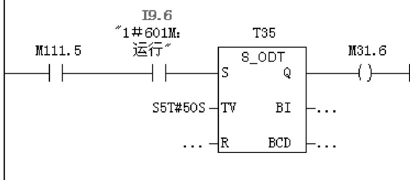



a)

图 11-45　主站泵阀控制子程序

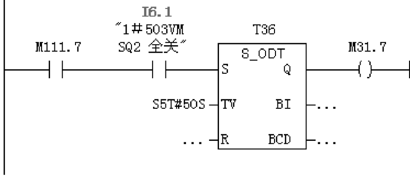
**程序段?12**：标题：

M31.6 泵阀连锁开泵开阀点



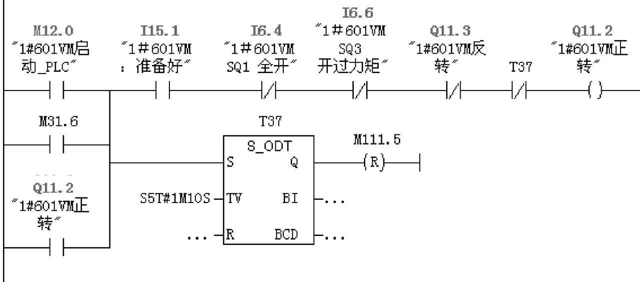

**程序段?13**：标题：

M31.7 泵阀连锁关阀关泵点




**程序段?14**：标题：

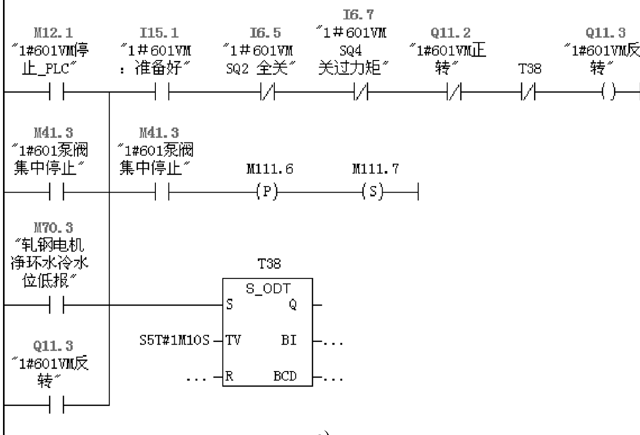
主站601VM电动蝶阀 开阀控制
M12.0 PLC手动正转（开阀）




b)

**程序段?15**：标题：

主站601VM电动蝶阀 关阀控制
M12.1 PLC手动反转（关阀）




c)

图 11-45 主站泵阀控制子程序（续）

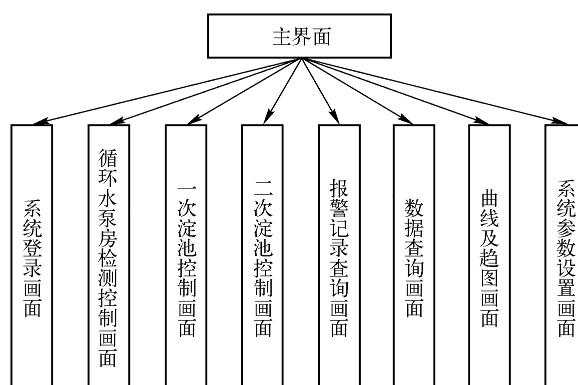



图 11-46　界面总体框架图

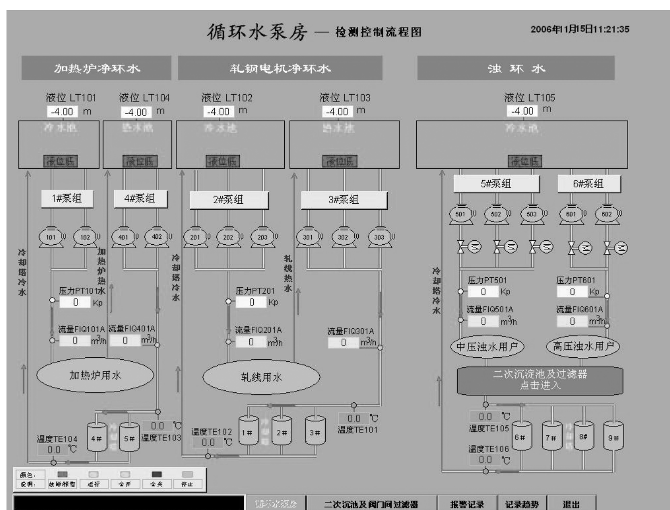



图 11-47　循环水泵房的检测控制画面

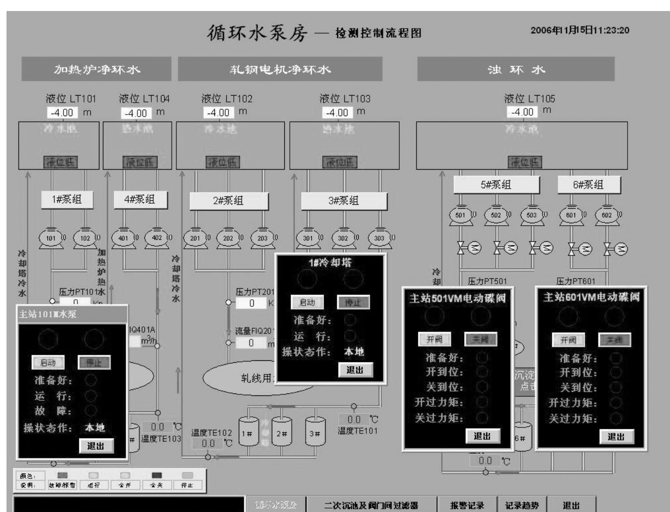



图 11-48　循环水泵房检测流程图中的部分泵阀控制界面

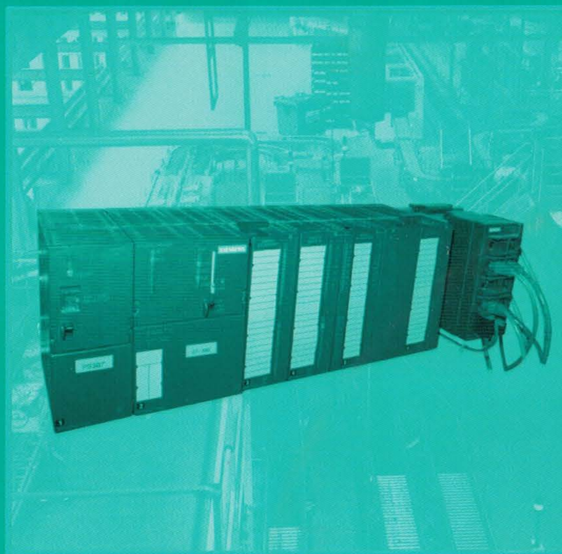
# 参 考 文 献


[1] Training Material, 2nd Siemens Automation & Drives Summer School at University of Applied Sciences, Esslingen FHTE, 2005.
[2] A1482, Beijing: Siemens Automation Training Center, 2008.
[3] 廖常初. S7-300/400 PLC 应用技术[M]. 北京：机械工业出版社，2005.
[4] 汪志锋. 可编程序控制器原理与应用[M]. 西安：西安电子科技大学出版社，2004.
[5] 王兆义. 可编程控制器教程[M]. 北京：机械工业出版社，2000.
[6] 周万珍，高鸿斌. PLC 分析与设计应用[M]. 北京：电子工业出版社，2004.
[7] 张浩，谭克勤，朱守云. 现场总线与工业以太网络应用技术手册[M]. 上海：上海科学技术出版社，2002.
[8] 崔坚. 西门子工业网络通信指南：上册[M]. 北京：机械工业出版社，2005.
[9] 崔坚. 西门子工业网络通信指南：下册[M]. 北京：机械工业出版社，2005.
[10] 苏昆哲，何华. 深入浅出西门子 WinCC V6[M]. 北京：北京航空航天大学出版社，2004.
[11] 刘锴，周海，等. 深入浅出西门子 S7-300 PLC[M]. 北京：北京航空航天大学出版社，2004.
[12] 姜建芳. 西门子 S7-200PLC 工程应用技术教程[M]. 北京：机械工业出版社，2010.
[13] 向晓汉. S7-300/400 PLC 基础与案例精选[M]. 北京：机械工业出版社，2011.

ISBN 978-7-111-36617-1
封面设计：孑时文化 ZiShi Culture

## 西门子工业自动化系列教材

- 工业自动化技术　陈瑞阳
- 过程自动化工程　孙洪程
- 西门子 S7-200 PLC 工程应用技术教程　姜建芳
- 西门子 S7-1200 PLC 编程与应用　刘华波
- 西门子 S7-300 PLC 工程应用技术教程　姜建芳
- S7-300/400 PLC 应用教程　廖常初
- 西门子 S7-300/400PLC 编程技术及工程应用　陈海霞
- 西门子 S7-300/400 PLC 编程与应用　刘华波
- 西门子人机界面（触摸屏）组态与应用技术 第 2 版　廖常初
- 人机界面组态与应用技术　席　巍





地址：北京市百万庄大街22号
电话服务
社服中心：(010)88361066
销售一部：(010)68326294
销售二部：(010)88379649
读者购书热线：(010)88379203

邮政编码：100037
网络服务
门户网：http:www.cmpbook.com
教材网：http:www.cmpedu.com
封面无防伪标均为盗版

定价：59.80元(含1DVD)

上架建议 工业技术/电气工程

ISBN 978-7-111-36617-1

